THE UNIVERSITY
of EDINBURGH

Classical secure delegation of quantum computations

Alexandru Cojocaru

Doctor of Philosophy
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh
2021

Abstract

The rapid evolution of quantum technologies is likely to cause major shifts in the mainstream computing landscape. In order to fully reach their potential in a wide base accessible to any user, remote access of quantum computers and manipulation of data with strong privacy and integrity guarantees are essential.

Consider a setting where a client having a fully classical computer wants to determine the result of some quantum computation, but lacks the necessary resources to perform the computation herself. She has access to a more powerful server which has quantum resources and can solve the problem and send the outcome back to the client. However, the client does not trust the powerful server, so she needs to find a way to hide her data. Therefore, the main question that arises is how can we guarantee the client's privacy of the input and even the computation itself against the server possessing quantum computational capabilities.

In the present thesis, we study this problem, denoted here as *classical secure delegation of quantum computations* (CSDQC) between a fully classical honest client and a quantum untrusted server. We focus on different models of security, analyzing the limitations and potential of each of the settings. Concretely, we first study the CSDQC problem under information-theoretic security. We analyse two categories of quantum computations, decision and sampling problems and in both cases we provide evidence indicating the impossibility of achieving information-theoretic security. Subsequently, we consider relaxing the security framework and specifically, we will analyze this task in the computational security setting (against quantum polynomial-time adversaries). As a result, in the second part of the thesis we put forward the *remote state preparation* as a key component that would allow us to achieve classical secure delegation of universal quantum computations. We present two protocols realizing the remote state preparation primitive assuming only a classical channel between client and server. The first candidate is shown to be secure in the honest-but-curious model, while the second candidate is proven secure against the server in the malicious setting. The security of both constructions relies on the hardness of the learning with errors problem. Finally, given the important role the remote state preparation plays not only in CSDQC, but also in other quantum communication protocols, we analyze its composable security to determine the privacy loss as a result of using remote state preparation as a sub-module in different protocols.

Acknowledgements

I would like here to show my appreciation to a part of those without whom the development of the current thesis could not be possible.

Firstly, I would like to express my gratitude to my main supervisor, Elham Kashefi, for the close guidance, continuous support, and for all the opportunities received throughout the whole period of my PhD.

I am extremely thankful to my co-supervisors, Petros Wallden and Aggelos Kiayias, for the entire support and scientific guidance and for the numerous discussions and patience to advise me this entire time.

I would also like to thank my two examiners, Frédéric Dupuis and Vesselin Velichkov for their extremely helpful feedback and for the valuable discussions during the PhD examination. I am also thankful to Myrto Arapinis for the very useful feedback during all my annual reviews.

During my years in Edinburgh I have gained many strong friendships that I am highly grateful for. I would like to show my appreciation for all the beautiful moments and for all the discussions, to my close friends from the quantum group (former and current members): Andru, Atul, Léo, Mina, Dan, Dominik, Brian, Mahshid, Niraj, to my close friends from the crypto group: Thomas, Hendrik, Yiannis, to my office mates and to my friends from the football group. I would also like to thank my close friends from Romania, Ionuț and Alexandru.

Finally, I would like to thank my family for their uninterrupted support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

The ideas presented in the current thesis are based on the following papers:

1. Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. Complexity theoretic limitations on blind delegated quantum computation. In 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, pages 6:1-6:13.

2. Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. On the possibility of classical client blind quantum computing. Paper presented at 8th International Conference on Quantum Cryptography (QCrypt) 2018.

3. Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. QFactory: Classically instructed remote secret qubits preparation. In Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, pages 615-645.

4. Christian Badertscher, Alexandru Cojocaru, Léo Colisson, Elham Kashefi, Dominik Leichtle, Atul Mantri, Petros Wallden. Security Limitations of Classical-Client Delegated Quantum Computing. In Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security.

Additional papers, dealing with problems independent of the ones studied in the current dissertation, are listed below, and they are not included in the current text:

1. Alexandru Cojocaru, Juan A. Garay, Aggelos Kiayias, Fang Song, and Petros Wallden. The bitcoin backbone protocol against quantum adversaries. IACR Cryptology ePrint Archive, 2019:1150, 2019 ([CGK+19]).

2. Michele Ciampi, Alexandru Cojocaru, Elham Kashefi, Atul Mantri. Secure Quantum Two-Party Computation: Impossibility and Constructions. IACR Cryptology ePrint Archive, 2020:1286, 2020 ([CCKM20]).

(*Alexandru Cojocaru*)

In Memoriam

Ilie Petra

(1937 - 2017)

Cojocaru Viorel

(1960 - 2018)

Pentru familia mea

Table of Contents

Chapter 1

Introduction

The evolution of traditional computers is challenged by a series of technological limitations. Since the power of these machines is proportional to the number of transistors they have, the trend has been to construct smaller and more efficient components, until reaching computer constituents of size no bigger than an atom. The most important aspect that emerges is that at this scale we cannot use anymore the laws of classical physics to describe the behaviour of these devices. Instead, we need to use the theories which govern the microscopic world. Quantum computers arose from the idea of modelling any natural phenomena using the laws of quantum mechanics. Therefore, in order to develop more powerful computing devices, comes the requirement to harness quantum properties of atoms, which would provide the basis for the memory and processor of the new machine.

The rapid development of quantum technologies has increased the computational capacity of quantum servers. We can expect to see quantum devices with high variability in terms of architectures and capacities, the so-called noisy, intermediate-scale quantum (NISQ) devices [Pre18] (e.g. superconducting such as the devices developed by IBM, Rigetti, Google, IonQ) that are currently available to users via classical cloud platforms. Quantum computers will tackle problems in fields such as medical research, data analytics, machine learning, where the protection of sensitive data is a must. Therefore, in order to be able to proceed to the next milestone for the utility of these devices in a wider industrial base, the issues of privacy and integrity of the data manipulation must be addressed.

This raised the necessity of privacy preserving functionalities such as the research developed around quantum computing on encrypted data. Clients, with devices as light as mobile phones, will want to use the services offered by quantum computation and

communication protocols, in a way that their privacy is guaranteed.

Importantly, the devices of clients willing to use these services may not have the capacity to support quantum communications. This can be either because their devices are fully classical or they are not connected to the newly developed, and still in its first stages, quantum communications network.

This technological challenge has lead to the formation of an entirely new research field on developing classical client - quantum server protocols for *delegated computing*.

1.1 Secure Delegated Computing

Secure delegated quantum computing is a two-party cryptographic primitive, where a computationally weak client wishes to delegate an arbitrary quantum computation to an untrusted quantum server in a privacy-preserving manner. Specifically, the two main properties that need to be satisfied are *correctness*: after the interaction with the quantum server, the weak client obtains the result of the desired quantum computation and *blindness*: a malicious server learns no information about the client's input, output and possibly even the computation itself, irrespective of how he deviates from the protocol specifications. For this reason the problem is also known as *blind quantum computing*.

In the classical world, this task has been studied for decades. The problem of secure delegation of computations was introduced by Feigenbaum in [Fei86], where we consider that the client has an input x and would like to use the power of an untrusted powerful server to obtain the outcome of a function f applied on his data x. To solve this task, the client transforms x into another instance x', while server computes $f(x')$ and sends it back to client. The instance x' must satisfy the following properties. The client can efficiently compute $f(x)$ from $f(x')$ while x' does not reveal any information about x to the server. It was shown that such a scheme can be constructed for the discrete logarithm problem, but cannot be extended in a straightforward manner to a general class of problems (for instance, the author conjectures there cannot exist this type of encryption for the integer factoring problem). Secure delegated computation was further studied by Abadi et al [AFK87]. The authors defined a general framework for this problem called Generalised Encryption Scheme (GES), which can be described at a high level as follows. Client and server interact for polynomially (in the size of input) many rounds and at the end, the client applies a decryption function and obtains the correct result $f(x)$ with a probability which is inverse polynomially in the

input size better than a random guess. These family of protocols are required to be information-theoretically secure, such that the server would learn at most the size of the input from the interaction with the client. Using this framework, they proved that any problem must belong to a certain advice complexity class and furthermore they showed that no $\mathsf{NP} - Hard$ problem can be securely delegated, unless the polynomial hierarchy collapses at the third level. In our work, in Chapter 3, we will use this framework to study the possibility of classical information-theoretically secure delegation of quantum computations.

A very important step in the study of delegated computation is represented by the introduction of homomorphic encryption (initially called privacy homomorphisms) in the seminal work of [RAD78]. This is a type of encryption allowing for function evaluation on top of the encrypted input, without using the secret key, which results in an encryption of the function evaluated on the plaintext. The homomorphic terminology comes from the fact that the encryption and decryption functions act as algebraic homomorphisms between the plaintext and ciphertext domains. Then, what the client needs to do is to generate a pair of public and secret key and encrypt using the public key her message x and send the encryption to the server. Then server will run the evaluation procedure for this function f on top of the encrypted input. Finally, the server sends the evaluation outcome to the client who decrypts the message using her secret key and obtains the desired $f(x)$. In the first stages, partially homomorphic encryptions that could evaluate only one type of operations (such as addition or multiplication) were constructed. Then, the first fully homomorphic encryption scheme (FHE), which allows for the evaluation of both additions and multiplications on top of ciphertexts, and hence for arbitrary circuits, was developed in the breakthrough result of Gentry [Gen09]. It is important to emphasize that the approaches of GES and FHE differ in two main aspects: i) FHE is a non-interactive protocol (after one round of communication, the client can obtain $f(x)$), while the GES allows for a polynomial (in the input size) number of messages exchanged between the 2 parties, ii) while GES requires information-theoretic security most of the FHE candidates are using only computational security, meaning that the privacy against the server is based on hardness assumptions (e.g. intractability of solving certain lattice problems).

The capabilities of the emerging quantum technologies, brought a raising interest in studying the secure delegation of computations problem when the server owns a quantum computer. On one hand, this brings the possibility of evaluating complex problems in a shorter time on the server side. On the other hand, quantum computing, which

equips attackers with unprecedented power, is changing the landscape of cryptography, with the known devastating consequences: Shor's quantum algorithm [Sho97], for example, solves factorization and discrete logarithm efficiently, and hence breaks the popular public-key cryptosystems based on them.

1.1.1 Semi-classical Client and Quantum Server

The problem of delegating quantum computations to an untrusted quantum server is also known as *blind quantum computing*, where, as explained above, the *blindness* refers to the privacy of the protocol, imposing that the server cannot infer anything about client's input, output and the underlying computation. Apart from the blindness, other desired properties are: *universality* - the protocol allows the delegation of any arbitrary quantum computation, *efficient interaction* - the client and server must interact for at most a polynomial (in the input size) number of messages (that can be either classical or quantum). Moreover, the blindness condition can also have different flavours: *information-theoretic* (or unconditional security) - the privacy of the input and computation holds irrespective of the power of the adversary, *computational* - the privacy holds only against quantum polynomial-time adversaries. As an additional property on top of either of these two blindness notions, the security can be *composable* - referring to the ability to use the delegated computation protocol in a secure manner as part of a larger context (which can be as simple as running in parallel two instances of the protocol).

The first-generation of blind quantum computation protocols are considering that the client is also quantum, but usually has only a small quantum device that gives her limited quantum capabilities, while the computation burden is still on server's side.

The first proposal of a secure delegated quantum computation protocol was introduced in [Chi05]. This construction can delegate universal quantum computations and achieves unconditional security against the quantum server, but assumes the client has a quantum memory (the capacity to store quantum states) and can apply a specific subset of quantum gates. Additionally, the protocol requires bidirectional quantum communication.

After this, there has been a large body of research exploiting the client-server setting defined in [Chi05], in order to relax the quantum computation and communication complexity of the client. In a following work of [AS03], the authors present a protocol for unconditional secure delegation for a subset of quantum computations denoted as

random verifiable functions (which are problems for which there exists an efficient algorithm to produce random input-output pairs). This protocol also requires quite heavy quantum abilities on client's side: client must sent multi-qubit states to the server and must be able to perform quantum measurements. The first proposal of a protocol that achieved blind quantum computations for universal computations, under unconditional security, while reducing significantly the quantum burden at a minimum for the client is the Universal Blind Quantum Computation protocol (UBQC) [BFK09]. This protocol is based on the measurement based quantum computation model [Joz05] and requires no quantum memory for the client, but just assumes that she has the ability to prepare and send single qubit states to the server in the first stage of the protocol, the rest of the communication being classical. Using the same computational model and same assumption for the client, the protocol proposed in [MF12] improves the efficiency of the UBQC scheme. Subsequently, a different type of relaxation of the client's quantum requirements was considered in [MF13], where instead of preparing and sending qubits, the client only needs to be able to measure the single qubit states sent by the server.

The composable security of general delegated quantum computation protocols, including the two previously mentioned protocols, was studied in [DFPR14] using the Constructive Cryptography (also known as Abstract Cryptography) framework [MR11], in order to analyse the security of this primitive in an arbitrary environment. The authors model the task of delegated quantum computation in a generic fashion independent of protocol requirements, specifications and universality of computations and provide a definition of blindness in the composable framework. Moreover, they show that the protocols of [BFK09] and [MF13] already satisfy the definition of composable blindness. In a following work in [MK13a], it is also studied the composable security of the protocol of [MF13], showing that this construction achieves a stronger notion of composable blindness. The efficiency of blind quantum computation protocols in terms of quantum communication has been studied in [MPDF13, GMMR13, PDF15], where these works also provide optimized blind quantum computing schemes by using different encodings on the client side in order to reduce the quantum communication. Delegated quantum computing has also been analysed using different physical systems in [DKL11, Mor12, LDT$^+$18] and using different quantum resources [MDK15]. Furthermore, given the relatively small quantum requirements for the client, protocols such as the ones in [BFK09] and [MF12] respectively, have also been experimentally realized in optics experiments in [BKB$^+$12]

and in [GRB+16] respectively. Additional different proposals towards minimising the requirements on the client side from a quantum computation or communication perspective have been studied. In [Bro15a] a secure delegation quantum computing with information-theoretic security (based on quantum one-time pad) is constructed which requires quantum communication only for a subset of gates (non-Clifford gates) from the quantum circuit representing the target quantum computation. Another approach considered in [DK16], uses as a starting point the UBQC protocol [BFK09] and aims at determining the least conditions needed to establish the required correlations between client and server in UBQC. Through a series of reductions they show using the composable security framework [MR11], that composable blind quantum computation can be achieved while only requiring the client to prepare and send two pure states, with an arbitrarily high overlap. In the recent work of [Zha20], the author proposes a construction in which the quantum communication is now independent of the size of the computation and depends only (polynomially) on the security parameter (and where the size of the quantum circuit the client has to perform, is also polynomial in the security parameter). This property referred here as "succint" complexity for the client is achieved in a protocol whose security is proven in the quantum random oracle model. For more details on the topic of blind quantum computation we refer the readers to the review of this field in [Fit17].

Given the extensive research focusing on the practicality aspect of quantum delegated computation protocols (and related functionalities), another important direction considered was reducing the required communications by exploiting classical fully-homomorphic-encryption schemes [BJ15, DSS16], or by defining quantum fully homomorphic encryptions (QFHE) as quantum analogues of the FHE [Lia15, OTF15, TKO+16, TOR18, LC18]. In more details, [BJ15] presents a quantum homomorphic encryption scheme with a quantum client, that is efficient for a class of quantum computations (containing Clifford gates), but where the complexity of the decryption scales with the number of non-Clifford gates. The security of this scheme is computational as the construction relies on using as a sub-module a classical post-quantum secure FHE. In a subsequent work of [DSS16], building on the ideas of [BJ15], the authors propose an alternative quantum client QFHE with computational security (as it also relies on a classical post-quantum FHE), that is efficient even for circuits with non-Clifford gates. This comes at the cost of requiring the evaluation key (sent by client to the server) to contain one quantum gadget per non-Clifford gate, where the size of the gadget scales with the space complexity of the decryption function of the classical FHE primitive.

On the other hand, there has also been a broad research on information-theoretic secure QFHE. In [OTF15], using quantum error correction codes, it is constructed an information-theoretic QFHE scheme for quantum circuits having only a constant number of non-Clifford gates and where the computational complexity is similar to [BJ15]. Alongside these positive results, limitations towards achieving information-theoretic QFHE for universal quantum computations have also been proven. Specifically, the work of [YPDF14] showed that achieving deterministic QFHE with perfect security for an arbitrary quantum computation, would imply that the size of the encryption is exponential in the input size, hence efficient deterministic QFHE cannot exist. This impossibility result was further strengthen in [LC18], where it was shown that the no-go holds also for more general information-theoretic security (even when some security error is allowed). On the positive side, the authors propose a QHE scheme for a non-universal class of computations (known as IQP). Independently, through a different proof technique based on quantum error correction codes this impossibility result was also shown in [NS18].

However, in all these approaches, the users and providers do have access to quantum resources to achieve their goals, in particular to quantum channels in addition to classical communication channels. This requirement might prove to be challenging as it allows the access for quantum cloud services only to users with suitable quantum devices.

1.1.2 Fully-classical Client and Quantum Server

As all the families of schemes enumerated above require a reliable long-distance quantum communication network, connecting all the interested parties remains a challenging task. Communication via quantum channels is typically assumed such that the client can establish the necessary correlations with the server to securely delegate a quantum computation. This has the downside that all these protocols cannot be put to work for the average user unless a reliable quantum network is deployed. This lead to the main open problem in the field of whether classical client secure delegated quantum computation (CSDQC) can be achieved.

1.1.2.1 Information-Theoretic Security

Roughly speaking, a protocol is information-theoretic secure when its security holds against an adversary with unlimited computational power. The reason is that this no-

tion of security is based on the fundamental theorems of quantum physics instead of difficult mathematical computations, which is the case for computational security.

Answering the question of whether information-theoretic secure delegation of quantum computations from a fully classical client can be achieved has very non-trivial consequences in complexity theory.

In the information-theoretic setting (where the leakage to the adversary is at most the input size), the problem of CSDQC was first considered in [MK14]. The authors showed a negative result for a particular class of protocols. Namely, they show that if the protocol contains a single round of communication, where both the encryption and decryption are deterministic algorithms, then the existence of such a scheme achieving correctness and perfect blindness would imply that *BQP* is included in *NP*. This question has also been posed in [DK16], where it was suggested the use of the above described classical Generalied Encryption Sheme (GES), in order to analyse the complexity theoretic implications of this task. In Chapter 3 ([ACGK19]) we show that the existence of general protocols achieving information-theoretic classical delegation is unlikely, by presenting an oracle separation between BQP and the class of problems that can be solved using a GES ([AFK87]).

While these results indicate restrictions on which of the above properties are jointly achievable for classical clients, completing the picture of CSDQC remains an open problem. In the light of these evidences, the natural direction would be exploring the task of classical delegation under a weaker security notion, namely computational security.

1.1.2.2 Computational Security

The above described results indicate restrictions on the task of classical secure delegation of quantum computations in the information-theoretic secure setting. Therefore, the expected path would be to consider fully-classical client solutions ensuring more restricted levels of security.

The first procedure exploring the possibility of CSDQC was proposed in [MDMF17]. The authors introduce a protocol in the measurement based quantum computing (MBQC) model, where the idea is to exploit the fact that in this computational model there can exist different computations requiring the same classical communication and having the same outcome. This property denoted as “flow ambiguity” results in some weaker notion of blindness (not for universal computations), allowing to partially hide the description of the computation from the server.

Until now the most promising solution for CSDQC is represented by protocols offering *post-quantum computational security* (security against quantum polynomial-time adversaries). These solutions can be categorized in two families: FHE-based candidates and protocols relying on the remote state preparation primitive, and we will describe both in details below.

1.1.2.2.1 FHE approach Following this path, classical client non-interactive blind delegated computing can be achieved for universal quantum computations. In a recent breakthrough of [Mah18], it was presented the first such classical client FHE procedure for quantum computations achieving computational security against the remote untrusted server. It was shown that solving this problem reduces to the server performing a controlled-CNOT gate given a classical encryption of the control bit. Then, this task can be achieved using a pair of post-quantum trapdoor claw-free functions (pair of injective functions having the same image, which are easy to invert using some secret key, but without this key it is hard even for a quantum computer to find a pair of inputs - known as claw - mapped by the two functions to the same value) requiring with some additional properties (e.g. the xor of the last bits of the elements of each claw is equal to the aforementioned control bit). Additionally, in order for the server to perform the quantum operations homomorphically, the classical encryption (used for encrypting the control bit) is required to be "quantum capable", meaning it needs to be a homomorphic encryption with extra conditions (such as randomness recoverability and invariance of the ciphertext form). Finally, the author shows how all these primitives can be realized, by giving a construction relying on a particular FHE scheme. This classical FHE for quantum computations was later followed by the work of [Bra18], where the construction achieved stronger security guarantee and relying on more standard post-quantum cryptographic assumptions.

1.1.2.2.2 Remote State Preparation One of the central building blocks in removing the need for quantum communication in a delegated quantum computing protocol is secure *remote state preparation* (RSP). This notion was initially introduced in [DKL11], in order to weaken the requirements on the client's side in the UBQC protocol. More specifically, as the perfect generation of $|+_\theta\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$ states is difficult from a practical point of view, they propose a single qubit preparation procedure where the client is using the polarization of weak coherent pulses sent over a lossy quantum channel.

At a high level, RSP resources allow a client to remotely prepare a quantum state on the server and consequently, they can be seen as the natural toolbox to replace quantum communication in a modular way. Moreover, from a security point of view, the RSP resources appear to enable a large family of composable protocols [DKL11, DFPR14, BFK09]. The importance of the classical RSP primitive used as a sub-module of larger protocols, due to its role in replacing quantum channels, stems from their ability to make quantum communication and computation protocols available to classical users, in particular clients without quantum-capable infrastructure on their end.

Motivated by this practical constrain, we introduce the first protocol mimicking this remote state preparation resource over a purely *classical* channel in Chapter 4 ([CCKW18]) under the assumption that the Learning-With-Errors (LWE) problem is computationally hard for quantum servers. This is a cryptographic primitive between a fully classical client and a server owning a quantum computer. By the end of the protocol the client has "prepared" remotely on the server's lab, a quantum state (typically a single qubit $|+_\theta\rangle$). This protocol further enjoys some important privacy guarantees with respect to the prepared state. Similar to the QFHE approaches of [Mah18] and [Bra18], our independent work, is also achieving post-quantum computational security, taking a different approach, more natural to measurement-based quantum computing protocols. Then, it can be observed that classical secure delegation of quantum computations can be obtained by combining for instance the UBQC protocol with this remote state preparation primitive, whose purpose is to eliminate all the quantum requirements of the client in the UBQC protocol. As a result, our solution for the CSDQC problem is an interactive protocol, while the protocols of [Mah18], [Bra18] are non-interactive. However, the approach we take is modular, while these constructions, proved the desired properties in a monolithic way. Specifically, as mentioned above, our RSP solution replaces the quantum channel (that is used in many different protocol implementing blind quantum computation) with a computationally (but post-quantum) secure generation of secret and random qubits (running between a classical client and a quantum server). This can be used by classical clients to achieve blind quantum computing but also, because of the modularity of the functionality, can be used in a number of other applications or functionalities (such as multi-party quantum computation [CCKM20]).

However, in [CCKW18] the security proof was shown in a weak "honest-but-curious" model, and the full proof of security was left as an open question. In Chapter 5 ([CCKW19]) we manage to further simplify the classical RSP in terms of the core

functionality (the set of produced states are the BB84 quantum states $\{|0\rangle, |1\rangle, |+\rangle := 1/\sqrt{2}(|0\rangle + |1\rangle), |-\rangle := 1/\sqrt{2}(|0\rangle - |1\rangle)\}$), while obtaining the security in the most general adversarial setting (fully malicious adversary) at the module level. As before, all our proofs are made using reductions to hardness assumptions (namely the LWE problem), and the simplicity of the protocol indicates that it is possible to perform an analysis of this module in a composable model such as Constructive Cryptography (CC) [MR11]. Very importantly, in [BCC+20], it was proven that this classical RSP called QFactory, when used as a subroutine of the UBQC protocol leads to classical secure delegation of universal quantum computations.

Concurrently, [GV19] gave another protocol that offers the stronger notion of *verifiable* classical RSP (the basic primitive they derive is a verifiable version of the one introduced in [CCKW18]) and proved the security of their primitive in the CC framework. However, their security analysis relies on an assumption called "Measurement Buffer" that forces the adversary to give the state that he is supposed to measure to the simulator, enforcing (essentially) a trusted measurement. In more details, the Measurement Buffer resource externalizes the measurement done by the distinguisher onto the simulator. In practice, this allows the simulator to change the state on the distinguisher side without letting him know. Intuitively, the Measurement Buffer recreates a quantum channel between the simulator and the server: when the simulator is not testing that the server is honest, the simulator replaces the state of the server with the quantum state sent by the ideal resource. In addition to [GV19], in [CCKW19] we also investigate the "abort" case of the protocol, which is related to the properties of the functions required for the protocol implementation. Specifically, these properties can only be achieved in a probabilistic fashion, causing the security of the protocol to fail whenever the function properties are not satisfied. By completely avoiding the abort scenario without changing the protocol, brings the downside of using less standard security parameters for the LWE hardness against the server (as explained in [Bra18]).

Using the Constructive Cryptography framework [MR11] is a common approach to analyze classical as well as quantum primitives and their composable security guarantees in general [DFPR14, DK16, MK13b].

Considering the importance of understanding the classical RSP primitive security when composed in larger contexts, using the CC framework, we study in Chapter 6 ([BCC+20]) the security loss incurred by using classical RSP as a sub-protocol in quantum communication and computation protocols. One of the main results of this work shows that any classical composable secure RSP will leak to the malicious server

the classical description of the produced quantum state. Additionally, we also show that some examples or RSP resources, such as verifiable RSP, are impossible to achieve using a reduction to the no-cloning theorem. This no-go result does not contradict the result of [GV19], but what our result shows is that it is impossible to realize this Measurement Buffer resource with a protocol interacting purely classically.

Recently, our classical RSP [CCKW19] was also used as a sub-module by [Zha20] to design a blind quantum computing scheme with a succinct quantum client.

In conclusion, all current solutions for classical RSP can replace the quantum channel in secure delegated quantum computing protocols (such as UBQC), but come at the cost of going from information-theoretic security using quantum communication to post-quantum computational security (and classical communication) via the described modules. The ultimate vision would be to develop a hybrid network of classical and quantum communication channels, depending on the desired security level and the technology development of quantum devices [WEH18].

The present dissertation deals with the problem of classical secure delegation of quantum computations (CSDQC). To briefly summarize the story of the thesis, we take the following path:

- We first investigate the possibility of information-theoretic secure CSDQC and provide complexity theoretic evidence that this is implausible to achieve;
- Next, we study this task under a weaker security notion, namely post-quantum computational security. As a first step we provide a solution which is secure only in the honest-but-curious security framework.
- Following, we present an improved solution which ensures security in the malicious framework against the adversarial quantum server.
- Finally, we analyze the composability property of our solutions when used as sub-protocols.

We expand more these contributions in the following section.

1.2 Contributions

1.2.1 Information-Theoretic Secure CSDQC

In the first part of the thesis we study classical delegation of quantum computations when the security guarantees that the quantum server learns nothing apart from the size of the computation, in an information-theoretic sense, problem denoted as ITS-CDQC. We perform this analysis using the Generalised Encryption Scheme (GES) framework of Abadi et al [AFK87]. This framework gives a complexity theoretic characterization of the class of problems that can be securely delegated, which allows us to investigate the possibility of ITS-CDQC by analyzing whether quantum polynomial-time computations can belong to this complexity class. We show that, provided certain complexity-theoretic conjectures are true, the power of ITS-CDQC is impossible to achieve.

Firstly, by considering that the interaction between client and server is bounded by a polynomial of fixed degree, we present an oracle relative to which a classical client is not able to information-theoretically secure delegate universal quantum decision problems. Specifically, if the client and the server exchange $O(n^d)$ bits of communication, this would imply that the class of problems that a quantum computer can solve in polynomial time (BQP) would be included in the advice complexity class $MA/O(n^d)$. We provide evidence that this containment is unlikely by proving that there exists an oracle relative to which $BQP \not\subset MA/O(n^d)$. The construction of the oracle providing this separation is based on the complement of Simon's problem and uses a diagonalisation argument. Our proof is incremental, we first show how to construct an oracle separating BQP and NP, following by a separation between BQP and MA. Next, we advance to the advice classes setting. We initially show how to construct an oracle relative to which BQP is not included in the class of problems that can be solved by deterministic polynomial-time algorithms receiving a bounded polynomial of advice, $\mathsf{P}/O(n^d)$. Finally, through a reduction to this separation, we show that for any degree d there exists an oracle with respect to which BQP is not a subset of $\mathsf{MA}/O(n^d)$.

Secondly, we show that if an ITS-CDQC protocol exists which allows the client to delegate quantum sampling problems, then there would exist circuits for computing the permanent of a matrix more efficiently than what is believed to be possible given the state of art for the complexity of solving the matrix permanent. Specifically, we prove that if the Boson Sampling problem can be delegated using an ITS-CDQC protocol, then there would exist circuits of size $2^{n-\Omega(n/log(n))}$ making polynomially-sized queries

to an $\mathsf{NP}^{\mathsf{NP}}$ oracle that can compute the permanent of an $n \times n$ matrix whose elements are in the set $\{-1,0,1\}$.

1.2.2 Computationally Secure CDQC against honest-but-curious server

In the *second part* we introduce the classical client remote state preparation primitive where a fully classical client can instruct the preparation of a sequence of random qubits at some distant party, i.e. untrusted quantum server. Their classical description is (computationally) unknown to any other party (including the distant party preparing them) but known to the client. We emphasize the crucial feature that no quantum communication is required to implement it. This primitive enables classical clients to participate in a wide range of quantum communication and computation protocols with only a public classical channel between the classical clients and the quantum server. A key such example is the delegated universal blind quantum computing problem, for example using our functionality one could achieve a purely classical-client computationally-secure delegated universal quantum computing.

We give a concrete protocol ($\mathsf{HBC-QFactory}$) implementing classical client remote state preparation, relying on the cryptographic primitive of trapdoor one-way function with certain additional properties (quantum-safe, two-regular, collision-resistant). The produced output states belong to the set $\{|+_\theta\rangle \,|\, \theta \in \{0,\pi/4,\cdots,7\pi/4\}\}$.

We then prove the security of $\mathsf{HBC-QFactory}$ in the Honest-But-Curious setting, which is a security model assuming that the malicious server needs to follow the protocol specifications, but he can use his classical information obtained from the protocol, in order to obtain any advantage in guessing the classical description of the quantum output [1]. Concretely, given that the produced state is of the form $|+_\theta\rangle$, the secret is represented by the classical description of this state, θ, which is unknown to the server, but known to the client using the trapdoor information. To show the security we prove that the classical description is a hardcore function and we show this by developing a similar reduction to the Goldreich-Levin Theorem.

Moreover, we provide methods for obtaining the required trapdoor functions from weaker assumptions: trapdoor permutation functions or homomorphic trapdoor functions. Then, to complete the construction of $\mathsf{HBC-QFactory}$, we present a family of

[1] Stronger notions of honest-but-curious in the quantum setting have been defined, such as the ones in [DNS10, SSS09].

functions relying on the LWE problem.

Finally, we give a proof-of-principle implementation of $\mathsf{HBC-QFactory}$ using the IBM quantum computer (IBM Quantum Experience). This experiment is completed using a toy trapdoor function, as due to the limited number of available qubits, we consider a simple function which cannot be post-quantum secure.

1.2.3 Computationally Secure CDQC against malicious server

After introducing the classical client remote state preparation primitive, in the *third part* we define a simpler (basic) primitive consisting of only BB84 states, and give a protocol called Malicious 4-states QFactory that implements this primitive and that is secure against the strongest possible adversary (an arbitrarily deviating malicious server).

The construction of Malicious 4-states QFactory relies on the following cryptographic primitives: quantum-safe, two-regular, collision resistant trapdoor one-way function and homomorphic hardcore predicate. The security of the protocol ensures that the basis of the generated qubits are completely hidden from any adversary and is based on the properties of the two families of functions.

The specific functions used are constructed based on known trapdoor one-way functions, resulting in the security of our basic primitive being reduced to the hardness of the Learning-With-Errors problem.

We then give a number of extensions of the Malicious 4-states QFactory protocol, demonstrating its modular construction. Firstly, we show an efficient secure extension to a classical client remote state preparation producing the 8 states $\{|+_\theta\rangle\}_\theta$. The security of this construction refers to the fact that the basis of the produced state is completely hidden. Finally, we give proper consideration of the abort case occurring when the properties of the trapdoor one-way functions are not satisfied.

1.2.4 Composable Security of CDQC

As indicated by the previous two protocols, remote state preparation realized using only a classical channel is one of the promising candidates to eliminate the need for quantum channels in several quantum communication and computation protocols. This primitive allows to rely solely on classical channels between client and server and yet benefit from its quantum capabilities while retaining privacy, because it enables a client, using only classical communication resources, to remotely prepare a quan-

tum state. In the *fourth part*, we analyze the security when employing classical-client remote state preparation ($\mathsf{RSP}_{\mathsf{CC}}$) as a sub-module in any protocol to avoid quantum channels. In this part, we investigate this question using the Constructive Cryptography framework [MR11] and we discover the security limitations of using $\mathsf{RSP}_{\mathsf{CC}}$ as a general sub-module but also in the context of classical delegation of quantum computations, when used inside the UBQC protocol.

Firstly, we prove a limitation specific to any composable classical client RSP, which stems from the following relation. If an RSP resource is realized by a $\mathsf{RSP}_{\mathsf{CC}}$ protocol with security against QPT distinguishers, then the resource will leak an encoded, but complete description of the underlying produced state. This connection between the composability of $\mathsf{RSP}_{\mathsf{CC}}$ (computational notion) and the statistical leakage of the ideal resource it is achieving (an information-theoretic notion) allows us to show that some desirable RSP resources are impossible to classically realize. Concretely, we show that if some specific verifiable RSP resource is computationally secure implementable then this would imply the existence of a quantum cloner.

Secondly, despite these security limitations, we might still be able to use a classical client RSP protocol as a sub-module in other specific protocol and expect the combined protocol as a whole to be composable secure. In this direction, we study the composable security of $\mathsf{UBQC}_{\mathsf{CC}}$, the family of protocols where an $\mathsf{RSP}_{\mathsf{CC}}$ is used for replacing the quantum channel required for the UBQC protocol, hence enabling fully classical clients. We prove that $\mathsf{UBQC}_{\mathsf{CC}}$ cannot be composable (even against QPT distinguishers) and to show this impossibility result we proceed in the following manner. Using a proof by contradiction we first show this task reduces to the existence of a composable single-qubit $\mathsf{UBQC}_{\mathsf{CC}}$ protocol (where the computation is described by a single qubit in the MBQC model). Then, we prove that the single-qubit UBQC resource can be turned into an RSP resource, which allows us to use the characterization we developed for RSP protocols. Finally, we show that the existence of such an RSP resource would violate the no-signalling principle.

1.3 Outline

In Chapter 2 we present the required definitions from quantum computing, complexity theory, security frameworks and cryptographic tools that we will use throughout the dissertation. Chapter 3 introduces the notions of information-theoretic secure classical delegation of quantum computations and the Generalised Encryption Scheme frame-

work. We present evidence for the no-go of this task in the case of decision problems in Chapter 3.1 and for sampling problems in Chapter 3.2.

In Chapter 4 we introduce the primitive remote state preparation constructed using a classical channel and we describe the protocol $\mathsf{HBC-QFactory}$ achieving this, whose security is proven in the honest-but-curious model through a Goldreich-Levin type of reduction in Chapter 4.4, assuming a certain family of post-quantum trapdoor one-way functions exist. Then we construct such a family of functions based on the Learning-With-Errors problem in Chapter 4.5 and finally we also present an implementation of this protocol using a toy function on the IBM quantum cloud service in Chapter 4.6.

Next, in Chapter 5 we present our protocol Malicious QFactory for classical client remote state preparation with enhanced security against any malicious server. The standard protocol is presented in Chapter 5.2 and similarly, the required underlying function construction is described Chapter 5.3. Additionally, the extension to a larger set of produced states is given in Chapter 5.4 and finally a complete analysis of the abort of the protocol caused by the probabilistic nature of the function properties required for the protocol construction is shown in Chapter 5.5.

In Chapter 6, we provide the characterization of classical-client remote state preparation resources through the lens of composable security framework. In Chapter 6.2 we show the impossibility of classically realizing remote state preparation, while in Chapter 6.3 we also prove that even in the context when a classical-client remote state preparation protocol is used to replace the quantum channel in a delegated quantum computation protocol, the result cannot be composable secure.

Finally, in Chapter 7 we conclude with a summary of our results and possible future directions.

Chapter 2

Preliminaries

2.1 Quantum Computing

We begin by introducing some quantum computing notions required for the understanding of our main results. For a more thorough description we recommend [NC00].

The central unit of quantum information and computation is the *qubit*, the quantum analogue of a classical bit. As classical bits can take 2 possible values, 0 or 1, the qubit analogous are the states $|0\rangle$ and $|1\rangle$. However, the qubit is more general, with the crucial difference is that a qubit can also act as if it is in the $|0\rangle$ and $|1\rangle$ states at the same time, more specifically as a linear combination of them. We can get a better intuition by considering the physical representation of a qubit: for a particle, its spin state can be aligned up (corresponding to state $|1\rangle$), down (corresponding to state $|0\rangle$) or be arbitrarily aligned, in between these 2 states (a linear combination of the up and down states). Therefore, a qubit, which we denote here as the state $|\psi\rangle$, can be expressed as: $|\psi\rangle = a|0\rangle + b|1\rangle$, where the coefficients $a,b \in \mathbb{C}$ satisfy $|a|^2 + |b|^2 = 1$ and are also known as *amplitudes* of the qubit $|\psi\rangle$.

Mathematically, we say that $|\psi\rangle$ is a vector in a complex Hilbert Space $\mathcal{H}$ of dimension 2. For instance, the quantum states $|0\rangle$ and $|1\rangle$ can be represented in vector form as: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, hence for a general qubit we have: $|\psi\rangle = a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

We also denote $\langle\psi|$ the Hermitian conjugate of $|\psi\rangle$: $\langle\psi| = \bar{a}\begin{pmatrix} 1 & 0 \end{pmatrix} + \bar{b}\begin{pmatrix} 0 & 1 \end{pmatrix}$.

Moreover, $|0\rangle$ and $|1\rangle$ form an *orthonormal* basis, known as the *computational basis*. Another examples of basis are the *Hadamard basis* described by the 2 quantum states $\{|+\rangle, |-\rangle\}$, where $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ and the *rotated Hadamard basis* $\{|+_\theta\rangle, |-_\theta\rangle\}$, where $|+_\theta\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{i\theta}|1\rangle$ and $|-_\theta\rangle =$

$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}e^{i\theta}|1\rangle$, and θ is an angle in $[0, 2\pi]$.

The evolution of quantum states can be described using *unitary operators*. An operator U is called unitary if and only if $UU^\dagger = I$, where $U^\dagger$ represents the hermitian conjugate of U and I denotes the identity matrix.

In order to describe quantum systems containing more than 1 qubit we use *tensor product* as a way to join multiple vector spaces. For instance if we have a composite system consisting of 2 single-qubit components $|\psi_1\rangle$ and $|\psi_2\rangle$, we denote the 2-qubit system as: $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. We will call $|\psi\rangle$ a *product state*.

However, not all composite quantum states can be represented as tensor products of quantum states. In those cases, we say the quantum state is *entangled*. One such example of state is the GHZ state: $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$.

In order to extract classical information from the quantum states we need to perform *quantum measurements*. The result of a quantum measurement is fundamentally random and to describe a measurement we use a collection measurement operators $\{M_1, ..., M_m\}$ each corresponding to a possible measurement result m. If the quantum system is described by the state $|\psi\rangle$, then by performing a quantum measurement, the probability of each outcome m, denoted by $p(m)$ is equal to: $p(m) \langle\psi| M_m^\dagger M_m |\psi\rangle$. If the measurement outcome was equal to m then after the measurement the state of the system becomes $\frac{1}{\sqrt{p(m)}} M_m |\psi\rangle$. In order to ensure that the sum of the probabilities of all measurement results is equal to 1, the measurements operator satisfy the *completeness* condition: $\sum_m M_m^\dagger M_m = I$.

An important class of quantum measurements are called *projective measurements*. In this case the measurement operators satisfy the following 2 conditions: M_m are hermitian ($M_m^\dagger = M_m$) and $M_i M_j = \delta_{i,j} M_i$. For the examples the measurements described by the sets of operators $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$, or $\{|+\rangle\langle +|, |-\rangle\langle -|\}$ or $\{|+_\theta\rangle\langle +_\theta|, |-_\theta\rangle\langle -_\theta|\}$ are all projective measurements.

Another representation of quantum states can be done using *density matrices*. We say a matrix ρ is a density matrix if it satisfies the following: $Tr(\rho) = 1$, it is hermitian and positive semi-definite. Therefore, if the quantum system is in the state $|\psi\rangle$, then we can represent it as the density operator $\rho = |\psi\rangle\langle\psi|$.

Until now, we have assumed that the state of a quantum system is always known to be in some state $|\psi\rangle$. In this case, we say the system is in a *pure state*. If the state of the system is however *unknown*, we say the system is in a *mixed state*. Specifically, if our quantum state can be one of the states $|\psi_i\rangle$, each with probability p_i, then these form an ensemble of pure states $\{p_i, |\psi_i\rangle\}_i$ and more importantly, we can represent the state

of the quantum system as the density operator: $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$.
Now using the density operator representation, we can also describe the before-mentioned state transformations. Namely, for quantum measurements described by a collection $\{M_i\}_i$, the probability to obtain outcome m is $p(m) = Tr(M_m^\dagger M_m \rho)$ and the state after obtaining result m becomes $\frac{1}{p(m)} M_m \rho M_m^\dagger$. And the unitary evolution is defined as: $U\rho U^\dagger$. However unitaries are not the most general way a quantum state can evolve. The most general family of transformations that can be applied to density operators are called *CPTP maps* (completely positive and trace preserving). We say a transformation Φ is a CPTP map if it acts on a state ρ as: $\Phi(\rho) = \sum_i K_i \rho K_i^\dagger$, such that $\sum_i K_i K_i^\dagger = I$. The linear operators K_i are known as *Kraus operators*. Using the density operator representation we can also describe the state of a subsystem of a composite state. Namely, if we have a system composing of 2 components denoted as ρ_{AB}, then the first subsystem can be described as $\rho_A = Tr_B(\rho_{AB})$ and the second subsystem as $\rho_B = Tr_A(\rho_{AB})$, where Tr_B and Tr_A denote the partial trace over the second and respectively first component. When studying the states of different quantum systems, in quantum cryptography we would like to analyze how far one state is from the other one, or how easy is to distinguish one quantum state from the other. To quantify how distinct two quantum states are we use the *trace distance*, which can be thought as a generalization of the *total variation distance*.The *total variation distance* between 2 probability distributions $\mathcal{D}_\infty$ and $\mathcal{D}_\in$ over a set S is defined as:

$$||\mathcal{D}_1 - \mathcal{D}_2|| = \frac{1}{2} \sum_{x \in S} |\mathcal{D}_1(x) - \mathcal{D}_2(x)| \tag{2.1}$$

Then, the trace distance between 2 states ρ_1 and ρ_2 is:

$$T(\rho_1, \rho_2) = \frac{1}{2} Tr\left(\sqrt{(\rho_1 - \rho_2)^2}\right) \tag{2.2}$$

The trace distance can also be interpreted as the maximum probability to distinguish between ρ_1 and ρ_2. Another useful measure of distance between 2 states ρ_1 and ρ_2 is called *fidelity*. When ρ_2 is a pure state $\rho_2 = |\psi_2\rangle\langle\psi_2|$, the fidelity is defined as: $F(\rho_1, |\psi_2\rangle\langle\psi_2|) = \langle\psi_2|\rho_1|\psi_2\rangle$.

In order to describe quantum computations we use the *quantum circuit model*, the quantum analogue of the classical circuit model. This model defines a mechanism to implement any possible quantum computation and specifically defines quantum gates that allow us to obtain quantum circuits manipulating quantum states. The most important quantum gates encountered in the remaining chapters are the following: $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ mapping $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$, $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$,

$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ mapping $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$, the Rotation around Z-axis $R_Z(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$, the 2-qubit CNOT gate: $CNOT(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |x \oplus y\rangle$.
Additionally, we can also define a quantum transformation based on a classical function.

Definition 2.1.1 (Function Unitary)**.** *For any function $f : A \to B$ that can be described by a polynomially-sized classical circuit, we define the controlled-unitary U_f, as acting in the following way:*

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle \;\; \forall\, x \in A \;\forall\, y \in B, \tag{2.3}$$

where we name the first register $|x\rangle$ control and the second register $|y\rangle$ target. Given the classical description of this function f, we can always define a QPT algorithm that efficiently implements U_f.

Definition 2.1.2 (Quantum Instrument)**.** *A map $\Lambda : \mathbb{C}^{n\times n} \to \{0,1\}^{m_1} \times \mathbb{C}^{m_2 \times m_2}$ is said to be a quantum instrument if there exists a collection $\{\mathcal{E}_y\}_{y\in\{0,1\}^{m_1}}$ of trace-non-increasing completely positive maps such that the sum is trace-preserving (i.e. for any positive operator ρ, $\sum_y \mathcal{E}_y(\rho) = \mathrm{Tr}(\rho)$), and, if we define $\rho_y = \frac{\mathcal{E}_y(\rho)}{\mathrm{Tr}(\mathcal{E}_y(\rho))}$, then $\Pr[\Lambda(\rho) = (y, \rho_y)] = \mathrm{Tr}(\mathcal{E}_y(\rho))$.*

2.2 Complexity Theory Background

In this section we cover some necessary background in Complexity Theory, essential for the understanding of the Chapter 3. For extended details, see [AB09, BDG12, BDG90].
We proceed by defining the complexity classes, both classical and quantum, which are going to be encountered in this thesis.

- P is the class of problems which can be solved efficiently in polynomial time by a deterministic Turing Machine [AB09]. We say that a language L belongs to the class P if we can build a deterministic Turing Machine M running in polynomial time, such that:
  - for every $x \in L$, M accepts x (returns 1)
  - for every $x \notin L$, M rejects x (returns 0).

- NP is the class of problems which can be computed in polynomial time by a nondeterministic Turing Machine [AB09]. Every NP algorithm has 2 stages: the first in which he makes a guess for the result of the problem and the second where he verifies in polynomial time if the guess was a correct answer. A useful way to describe the behaviour of any NP algorithm is by looking at its computational tree [BDG12]. The computational tree indicates how does the algorithm work given a specific input x to the problem. The initial configuration of the Turing Machine on the input x represents the root of the tree. Every internal node corresponds to a computation performed by the algorithm and the children of the node are the possible configurations which can be obtained in one computational step. For each nonterminal node, the choice for the next computation is made in a nondeterministic way. In a *decision* problem, the leaves of the computational tree are either accepting or rejecting states. Then, we say that a string s is accepted by the language defined in the decision problem, if in the computational tree generated for s there *exists at least one* path ending in an *accepting* state. On the other hand, a string s is rejected if *all possible* paths of the tree end in a *reject* state. A decision problem G belongs to the class coNP, if the complement of G, obtained by switching the accepting answers with the rejecting answers and vice versa, is in the NP class. In this way, we have that a string x is accepted if every path of the computational tree ends in an accepting state and is rejected if there exists one path ending in a rejecting state.
  Another definition for the class NP, which is going to be used in Chapter 3 is the following:
  A language L is in NP if there exists a polynomial-time TM V and a polynomial p such that for every $x \in \{0,1\}^*$, we have:

$$x \in L \text{ if and only if } \exists \text{ a } \textit{witness } \ w \in \{0,1\}^{p(|x|)} \text{ such that } M(x,w) = 1$$

  Observation: The class of NP Turing Machines is *countable*.

- The BPP class (Bounded-error Probabilistic Polynomial Time) contains the problems which can be solved by probabilistic Turing Machines in polynomial time, with the probability of giving a wrong answer being less than $1/3$. We say a language L is in the BPP class if we can construct a probabilistic Turing Machine M running in polynomial time such that:

  - If $x \in L$, then M accepts x with probability $p \geq 2/3$;

  - If $x \notin L$, then M accepts x with probability $p \leq 1/3$;

  Notice that the choice of the constant $1/3$ is arbitrary. In general, the error can be as high as $\frac{1}{2} - \frac{1}{n^c}$, for any positive constant c.

- The MA class (Merlin-Arthur) contains the problems that can be solved in polynomial time by Merlin-Arthur protocols. These are protocols where Merlin who is computationally unbounded sends to Arthur a polynomial-sized proof, while Arthur must verify this proof by running a probabilistic polynomial-time computation. We say a language L is in MA if there exists a polynomial time TM and a polynomial p such that:

  - If $x \in L$, then there exists a witness $w \in \{0,1\}^{p(|x|)}$ such that $M(x,w)$ accepts with probability at least $2/3$
  - If $x \notin L$, then for all witnesses $w \in \{0,1\}^{p(|x|)}$, $M(x,w)$ accepts with probability at most $1/3$.

- The BQP class (Bounded-error Quantum Polynomial Time) is the quantum equivalent of the BPP class. BQP consists of the problems which can be solved in polynomial time by Quantum Turing Machines with the probability of giving a wrong answer being less than $1/3$ [NC00, Aar10]. We can prove that a decision problem G belongs to BQP by showing that there exists a quantum circuit of polynomial size (as defined in the quantum circuit model) that can solve G with bounded-error.

Additionally, BQP is *closed under complement*, meaning that the complement of any problem in BQP also belongs to the class BQP.

A very important problem belonging to the class BQP which we will make use of in our work is the *Simon's problem*.
The input of Simon's problem is a function from n-bit strings to n-bit strings f, to which we only have "black-box" access (meaning that we cannot see the description of the function and we are only allowed to query it on elements of the domain and receive the value of the function on those points).
The input function $f : \{0,1\}^n \to \{0,1\}^n$ is guaranteed to satisfy one of the following 2 properties:

1. f is 1-to-1: f is a permutation of the set $\{0,1\}^n$.

2. f is 2-to-1: $\exists s \in \{0,1\}^n - \{0\}$ such that $\forall x \neq x'$ we have: $f(x) = f(x')$ if and only if $x' = x \oplus s$. We call s the xor mask of f.

The task of the problem is to determine whether the input f is 1-to-1 or 2-to-1.
To solve this problem, the optimal classical algorithm uses $O(2^{\frac{n}{2}})$ queries.
However, quantumly we can solve Simon's problem using $O(n)$ queries using Simon's algorithm [Sim97]. Therefore, both Simon's problem and its complement belong to the class BQP.

Decision Problems versus Sampling Problems. All the complexity classes described above refer to *decision problems* - problems in which the answer is either "yes" or "no" depending on whether an input x belongs or not to a language L. In this thesis, we also focus on *sampling problems* problems in which the answer is a sample from particular probability distribution. Specifically, we define a sampling problem S as a set of probability distributions $\{\mathcal{D}_x\}_x$ each corresponding to an input x, such that each $\mathcal{D}_x$ is a probability distribution over $\{0,1\}^{p(|x|)}$, for some polynomial p. Solving the sampling problem S essentially means that when receiving the input x, the task is to sample from the distribution $\mathcal{D}_x$, whereas solving S *approximately* means sampling from a distribution $\mathcal{D}'_x$, such that $||\mathcal{D}'_x - \mathcal{D}_x|| \leq \frac{1}{q(|x|)}$, for some polynomial q. Therefore, we can define SampBQP as the class of sampling problems for which there exists a quantum polynomial time algorithm that given as input a tuple (x, ε) will output a sample from a probability distribution $\mathcal{D}'_x$ satisfying $||\mathcal{D}'_x - \mathcal{D}_x|| \leq \varepsilon$. Such an example of SampBQP problem that we will encounter later in our work is the *Boson Sampling problem*.
Another class of problems which we will use in our work, is #P. This is neither a decision nor a sampling class, but is the class of all functions $f : \{0,1\}^* \to \mathbb{N}$ that take as input a description of a P algorithm and output the number of inputs which the algorithm can accept.

Now, we need to focus on 2 particular categories of problems: problems which can be solved by Turing Machines which receive an additional "help" called *advice* and problems which can be solved by Turing Machines which have access to an *oracle* that can correctly answer to particular questions. We are going to give a thorough description of these two models of computation as they are directly connected with our major results in Chapter 3.

2.2.1 Advice Turing Machines

An advice function can be thought of as any function $f : \mathbb{N} \to \Sigma^*$. Turing Machines with an advice function f receive an additional *help* to solve problems, in the form of an advice string $f(n)$. The most important aspect of the advice function is that this external information does not depend on the value of the input for a problem, but only on the input size n:
That is, for a given input x, the received advice is $f(|x|)$

Definition 2.2.1. *The class $\mathcal{C}/\mathcal{F}$ of languages recognized by Turing Machines with advice is the set of languages L associated with a language C from $\mathcal{C}$ together with an advice function f from $\mathcal{F}$ such that:*

$$A = \{x \in \Sigma^* \mid \textit{ the pair } (x, f(|x|)) \in C, C \in \mathcal{C}, f \in \mathcal{F}\} \tag{2.4}$$

Intuitively, we say that there exists a function f from $\mathcal{F}$, which gives the necessary additional information to a machine from the class $\mathcal{C}$ in order to accept a more difficult language belonging to the class $\mathcal{C}/\mathcal{F}$.
We will now indicate some commonly known classes of advice languages and their interpretation:

1. $\mathcal{F} = \log$ represents the set of functions f satisfying the property:
   $\forall\, n \in \mathbb{N}\ \exists\, c \in \mathbb{N}$ such that $|f(n)| \leq c \cdot \log(n)$. We define $\mathsf{P}/\log$ as the class of languages:
   $$L = \{x \,|\, (x, f(|x|)) \in A, A \in P, f \in log\} \tag{2.5}$$
   We say that for any language $L \in \mathsf{P}/\log$ we can determine if a string x belongs to L by using a deterministic polynomial time algorithm with the help of an external information s, where $|s| = \log(|x|)$.

2. $\mathcal{F} = \mathsf{poly}$ represents the family of functions f satisfying the property:
   $\forall\, n, \exists\, p$ a polynomial such that $|f(n)| \leq p(n)$. We define P/poly as the class of languages $L = \{x \,|\, (x, f(|x|)) \in A, A \in P, f \in \mathsf{poly}\}$.

3. NP/poly is represented by the set of languages:
   $$L = \{x \,|\, (x, f(|x|)) \in A, A \in NP, f \in \mathsf{poly}\} \tag{2.6}$$

Advice Turing Machines are known as *non-uniform* models, where we have a different algorithm for every possible size of the input, as opposed to the standard Turing

Machines known as *uniform* models where the same algorithm is used for all possible input lengths.

Another family of non-uniform models are *Boolean circuits*. A Boolean circuit computes a binary function by sequentially applying logical gates AND, OR, NOT on a given input.
We say a language L has polynomial circuits if there exists a family of circuits $\{C_n\}_n$ and a polynomial p such that:

1. The number of gates in the circuit is bounded by p: $|C_m| \leq p(m)$
2. For any $x \in \{0,1\}^m$ C_m receives m input bits and returns 1 if $x \in L$ and 0 otherwise.

The connection between Advice TM and Boolean circuits can be described in the following way. We say that a Boolean circuit family $\{C_n\}_n$ is f-size bounded if the number of gates of C_n is at most $f(n)$ for any n. And we will denote by $Size(f(n))$ the class of languages decided by $O(f(n))$-size bounded Boolean circuit family.
Then, the class $P/poly$ of determininstic TM that receive polynomial size advice, can also be described as: $P/poly = \cup_{d\geq 1} Size(n^d)$. In other words, we say a language L belongs to $P/poly$ if it can be solved by a polynomially-sized Boolean circuit family.

2.2.2 Oracle Turing Machines

Definition 2.2.2. *An oracle TM is a machine with access to an oracle that can decide if a string is in a language $O \subseteq \Sigma^*$.*

In addition to the standard TM, an *oracle* TM has an oracle tape which can be used to query for an extra input and the machine will receive in a single computational step an answer specifying whether that input belongs to the language O or not.
Any standard TM can be considered an oracle TM by setting the empty set $\emptyset$ as the oracle's language.

For any oracle machine M_O we are able to describe its *computational tree* in the following way. Let M_O run on input m. Then, we set the configuration of M_O on input m as the root of the computational tree. Next, every inner node of the tree represents a query made by M_O to the oracle O. For every such node we have 2 branches defining 2 different sets of computations which M_O might perform depending on the result of the query. Namely, if the queried string belongs to the language O, then we proceed to the computations specified in the left branch, otherwise to the computations indicated

by the right branch. In the nodes where no more queries are made (the leaves of the tree), M_O will take a decision, if it is an accepting state or not.

Now, we define the language L corresponding to the oracle O. For each input, we will obtain a single path π in the computation tree starting from the root and ending in an accept or reject node. For every node i the path goes to its left child if the string labelled at node i is inside the language L and goes to its right child otherwise. Finally, we conclude that the initial input m is in the language accepted by the oracle TM M_O if this resulting path π terminates in an accepting state.
Now, we will present 2 very important advice classes which we will also make use of in Chapter 3.

Definition 2.2.3. *We define* P^O *as the set of problems which can be solved by a deterministic polynomial-time TM which has access to the oracle O.*
Similarly, we define NP^O *as the set of problems solved by a nondeterministic polynomial-time TM with access to the oracle O.*

Whenever O is a language which cannot be decided by M, the oracle adds more computational power to M. For instance, the P^{SAT} represents the set of problems solved by a deterministic TM which also has access to an oracle capable of solving the SAT problem. This oracle will obviously help compute more functions.
Using oracles we can also define the *polynomial hierarchy (*PH*)*. We define the 0^{th} level of the polynomial hierarchy as: $\Sigma_0^{\mathsf{P}} = \mathsf{P}$ and $\Pi_0^{\mathsf{P}} = P$. we can define recursively the next levels of PH, in the following way: the k^{th} level can be computed as $\Sigma_k^{\mathsf{P}} = \mathsf{NP}^{\Sigma_{k-1}^{\mathsf{P}}}$ and $\Pi_k^{\mathsf{P}} = \mathsf{coNP}^{\Sigma_{k-1}^{\mathsf{P}}}$. And the PH is defined as: $\mathsf{PH} = \cup_{k\geq 0}\Sigma_k^{\mathsf{P}}$. Additionally, we say the PH collapses at the level k if $\Sigma_k^{\mathsf{P}} = \Pi_k^{\mathsf{P}}$.

A very important role of oracle classes, is that they allows us to identify relations between different complexity classes. Therefore, proving separations between classes of problem with respect to a given fixed oracle represents a strong evidence that those separations would also hold in the lack of the oracle.

2.3 Cryptographic Primitives

In this section, we are considering cryptographic primitives secure against quantum adversaries, so we assume that all the properties of our functions hold for a general Quantum Polynomial Time (QPT) adversary, rather than the usual Probabilistic Poly-

nomial Time (PPT) one. We will denote D the domain of the functions, while $D(n)$ is the subset of strings of length n.

Definition 2.3.1 (Quantum-Safe (informal))**.** *A protocol or a function is quantum-safe (also known as post-quantum secure), if all its properties remain valid when the adversaries are QPT (instead of PPT).*

The following definitions are for PPT adversaries, however in this paper we will generally use quantum-safe versions of those definitions and thus security is guaranteed against QPT adversaries.

Definition 2.3.2 (One-way)**.** *A family of functions* $\{f_k : D \to R\}_{k \in K}$ *is* ***one-way*** *if:*

- *There exists a PPT algorithm that can compute* $f_k(x)$ *for any index function k, outcome of the PPT parameter-generation algorithm Gen and any input* $x \in D$*;*
- *Any PPT algorithm* $\mathcal{A}$ *can invert* f_k *with at most negligible probability over the choice of k:*
  $$\Pr_{\substack{k \leftarrow Gen(1^n) \\ x \leftarrow D \\ rc \leftarrow \{0,1\}^*}} [f(\mathcal{A}(k, f_k(x)) = f(x)] \leq \mathsf{negl}(n)$$
  where rc represents the randomness used by $\mathcal{A}$

Definition 2.3.3 (Second preimage resistant)**.** *A family of functions* $\{f_k : D \to R\}_{k \in K}$ *is* ***second preimage resistant*** *if:*

- *There exists a PPT algorithm that can compute* $f_k(x)$ *for any index function k, outcome of the PPT parameter-generation algorithm Gen and any input* $x \in D$*;*
- *For any PPT algorithm* $\mathcal{A}$*, given an input x, it can find a different input* x' *such that* $f_k(x) = f_k(x')$ *with at most negligible probability over the choice of k:*
  $$\Pr_{\substack{k \leftarrow Gen(1^n) \\ x \leftarrow D \\ rc \leftarrow \{0,1\}^*}} [\mathcal{A}(k,x) = x' \text{such that } x \neq x' \text{ and } f_k(x) = f_k(x')] \leq \mathsf{negl}(n)$$
  where rc is the randomness of $\mathcal{A}$*;*

Definition 2.3.4 (Collision resistant)**.** *A family of functions* $\{f_k : D \to R\}_{k \in K}$ *is* ***collision resistant*** *if:*

- *There exists a PPT algorithm that can compute* $f_k(x)$ *for any index function k, outcome of the PPT parameter-generation algorithm Gen and any input* $x \in D$*;*

- *Any PPT algorithm $\mathcal{A}$ can find two inputs $x \neq x'$ such that $f_k(x) = f_k(x')$ with at most negligible probability over the choice of k:*
  $\Pr_{\substack{k \leftarrow Gen(1^n) \\ rc \leftarrow \{0,1\}^*}} [\mathcal{A}(k) = (x,x') \text{such that } x \neq x' \text{ and } f_k(x) = f_k(x')] \leq \mathsf{negl}(n)$
  where rc is the randomness of $\mathcal{A}$ (rc will be omitted from now).

Theorem 2.3.5. *[KL14] Any function that is collision resistant is also second preimage resistant.*

Definition 2.3.6 (k-regular)**.** *A deterministic function $f : D \to R$ is **k-regular** if $\forall y \in \mathrm{Im}(f)$, we have $|f^{-1}(y)| = k$.*

Definition 2.3.7 (Trapdoor Function)**.** *A family of functions $\{f_k : D \to R\}$ is a **trapdoor function** if:*

- *There exists a PPT algorithm* `Gen` *which on input 1^n outputs (k, t_k), where k represents the index of the function;*
- *$\{f_k : D \to R\}_{k \in K}$ is a family of one-way functions;*
- *There exists a PPT algorithm* `Inv`*, which on input t_k (which is called the trapdoor information) output by* `Gen`*(1^n) and $y = f_k(x)$ can invert y (by returning all preimages of y[1]) with non-negligible probability over the choice of (k, t_k) and uniform choice of x.*

Definition 2.3.8 (Hard-core Predicate)**.** *A function $hc : D \to \{0,1\}$ is a **hard-core predicate** for a function f if:*

- *There exists a QPT algorithm that for any input x can compute $hc(x)$;*
- *Any PPT algorithm $\mathcal{A}$ when given $f(x)$, can compute $hc(x)$ with negligible better than $1/2$ probability:*
  $\Pr_{\substack{x \leftarrow D(n) \\ rc \leftarrow \{0,1\}^*}} [\mathcal{A}(f(x), 1^n) = hc(x)] \leq \frac{1}{2} + \mathsf{negl}(n)$*, where rc represents the randomness used by $\mathcal{A}$;*

Definition 2.3.9 (Hard-core Function)**.** *A function $h : D \to E$ is a **hard-core function** for a function f if:*

[1] While in the standard definition of trapdoor functions it suffices for the inversion algorithm `Inv` to return one of the preimages of any output of the function, in our case we require a two-regular tradpdoor function where the inversion procedure returns both preimages for any function output.

- *There exists a QPT algorithm that can compute $h(x)$ for any input x*
- *For any PPT algorithm $\mathcal{A}$ when given $f(x)$, $\mathcal{A}$ can distinguish between $h(x)$ and a uniformly distributed element in E with at most negligible probability:*

$$\Big| \Pr_{x\leftarrow D(n)}[\mathcal{A}(f(x),h(x))=1] - \Pr_{\substack{x\leftarrow D(n)\\ r\leftarrow E(|h(x)|)}}[\mathcal{A}(f(x),r)=1]\Big| \leq \mathsf{negl}(n)$$

The intuition behind this definition is that as far as a QPT adversary is concerned, the hard-core function appears indistinguishable from a randomly chosen element of the same length.

Theorem 2.3.10 (Goldreich-Levin [GL89])**.** *From any one-way function $f\colon D\to R$, we can construct another one-way function $g\colon D\times D\to R\times D$ and a hard-core predicate for g. If f is a one-way function, then:*

- *$g(x,r)=(f(x),r)$ is a one-way function, where $|x|=|r|$.*
- *$hc(x,r)=\langle x,r\rangle \bmod 2$ is a hard-core predicate for g*

Informally, the Goldreich-Levin theorem is proving that when f is a one-way function, then $f(x)$ is hiding the xor of a random subset of bits of x from any PPT adversary[2].

Theorem 2.3.11 (Vazirani-Vazirani XOR-Condition Theorem [VV85])**.** *Function h is hard-core function for f if and only if the xor of any non-empty subset of h's bits is a hard-core predicate for f.*

2.4 Learning-With-Errors

The Learning-With-Errors problem (LWE) is described in the following way:

Definition 2.4.1 (LWE problem (informal))**.** *Given s, an n dimensional vector with elements in $\mathbb{Z}_q$, for some modulus q, the task is to distinguish between a set of polynomially many noisy random linear combinations of the elements of s and a set of polynomially many random numbers from $\mathbb{Z}_q$.*

[2]The Goldreich-Levin proof is using a reduction from breaking the hard-core predicate $hc(x,r)$ to breaking the one-wayness of h. In this thesis the functions we consider are one-way against quantum adversaries, and using the same reduction we conclude that $hc(x,r)$ is a hard-core predicate against QPT adversaries.

Regev [Reg05] and Peikert [Pei09] have given quantum and classical reductions from the average case of LWE to problems such as approximating the length of the shortest vector or the shortest independent vectors problem in the worst case, problems which are conjectured to be hard even for quantum computers.

Theorem 2.4.2 (Reduction LWE, from [Reg05, Therem 1.1])**.** *Let n, q be integers and $\alpha \in (0,1)$ be such that $\alpha q > 2\sqrt{n}$. If there exists an efficient algorithm that solves* $\mathrm{LWE}_{q,\bar{\Psi}_\alpha}$*, then there exists an efficient quantum algorithm that approximates the decision version of the shortest vector problem* GAPSVP *and the shortest independent vectors problem* SIVP *to within $\tilde{O}(n/\alpha)$ in the worst case.*

2.5 Constructive Cryptography Framework

The Constructive Cryptography (CC) framework (also sometimes referred to as the Abstract Cryptography (AC) framework) introduced by Maurer and Renner [MR11] is a top-down and axiomatic approach, where the desired functionality is described as an (ideal) *resource* $\mathcal{S}$ with a certain input-output behavior independent of any particular implementation scheme. A resource has some interfaces I corresponding to the different parties that could use the resource. In our case, we will have only two interfaces corresponding to Alice (the client) and Bob (the server), therefore $I = \{A, B\}$. Resources are not just used to describe the desired functionality (such as a perfect state preparation resource), but also to model the assumed resources of a protocol (for example a communication channel). The second important notion is the *converter* which is used to define a protocol. Converters always have two interfaces, an inner and an outer one, and the inner interface can be connected to the interface of a resource. For instance, if $\mathcal{R}$ is a resource and π_A, $\pi_B \in \Sigma$ are two converters (corresponding to a given protocol making use of resource $\mathcal{R}$) we can connect these two converters to the interface A and B, respectively, (the resulting object being a resource as well) using the following notation: $\pi_A \mathcal{R} \pi_B$. Furthermore, in this thesis we consider that any converter interacting classically on its inner interface and outputting a single quantum message on its outer interface can be represented as a sequence of quantum instruments (which is a generalization of CPTP maps taking into account both quantum and classical outputs, see Definition 2.1.2) and constitutes the most general expression of allowed quantum operations. More precisely, this model takes into account interactive converters (and models the computation in sequential dependent stages). This is sim-

ilar to the classical world where we would instantiate the converter by a sequence of classical Turing machines (passing states to each other) [Gol01].

A *filter* (usually denoted $\vdash$) is a special converter used to force a honest behaviour on a given interface of a resource. They are usually used to prove the correctness of a protocol, as they describe what can be done in an honest run. They are removed when we want to provide full power to a cheating adversary or to a simulator. Usually, in order to keep the filter simple, the functionality accepts as a first message a bit c which says if the party wants to behave honestly ($c = 0$) or maliciously ($c = 1$). That way, the filter $\vdash^{c=0}$ (or simply $\vdash$) just sends $c = 0$ to the resource, and then forwards all the messages between it's inner and outer interface.

In order to characterize the distance between two resources (and therefore the security), we use the so-called *distinguishers*. We then say that two resources $\mathcal{S}_1$ and $\mathcal{S}_2$ are indistinguishable (within ε), and denote it as $\mathcal{S}_1 \approx_\varepsilon \mathcal{S}_2$, if no distinguisher can distinguish between $\mathcal{S}_1$ and $\mathcal{S}_2$ with an advantage greater than ε. In this thesis, we will mostly focus on quantum-polynomial-time (QPT) distinguishers.

Central to Constructive Cryptography is the notion of a secure construction of an (ideal) resource $\mathcal{S}$ from an assumed resource $\mathcal{R}$ by a protocol (specified as a pair of converters). We directly state the definition for the special case we are interested in, namely in two-party protocols between a client A and a server B, where A is always considered to be honest. The definition can therefore be simplified as follows:

Definition 2.5.1 (See [Mau11, MR11])**.** *Let $I = \{A, B\}$ be a set of two interfaces (A being the left interface and B the right one), and let $\mathcal{R}, \mathcal{S}$ be two resources. Then, we say that for the two converters π_A, π_B, the protocol $\pi := (\pi_A, \pi_B)$* (securely) constructs *$\mathcal{S}$ from $\mathcal{R}$ within ε, or that $\mathcal{R}$* realizes *$\mathcal{S}$ within ε, denoted:*

$$\mathcal{R} \xrightarrow[\varepsilon]{\pi} \mathcal{S} \tag{2.7}$$

if the following two conditions are satisfied:

- *Availability (i.e. correctness):*

$$\pi_A \mathcal{R} \pi_B \approx_\varepsilon \mathcal{S} \vdash \tag{2.8}$$

  (where $\vdash$ represents a filter, i.e. a trivial converter that enforces honest/correct behavior [3], and $A \approx_\varepsilon B$ means that no polynomial quantum distinguisher can dis-

[3] Usually, a filter simply sends a bit $c = 0$ and then forwards all communications between its two interfaces (this filter will be denoted by $\vdash^{c=0}$), but it could be a more general converter. When the filter is not clear from the context, we need to specify also which filter we consider.

tinguish between A and B (given black-box access to A or B) with an advantage better than ε*)*

- *Security: there exists* $\sigma \in \Sigma$ *(called a simulator) such that:*

$$\pi_A \mathcal{R} \approx_\varepsilon \mathcal{S}\sigma \tag{2.9}$$

We also extend this definition when ε *is a function* $\varepsilon : \mathbb{N} \to \mathbb{R}$*: we say that* $\mathcal{S}$ *is* ε-realizable *if for any* $n \in \mathbb{N}$*,* $\mathcal{S}$ *is* $\varepsilon(n)$*-realizable.*

The intuition behind this definition is that if no distinguisher can know whether he is interacting with an ideal resource or with the real protocol, then it means that any attack done in the "real world" can also be done in the "ideal world". Because the ideal world is secure by definition, so is the real world. Using such a definition is particularly useful to capture the "leakage" of information to the server. This is quite to capture in the real world, but very natural in the ideal world.

Lemma 2.5.2 ([MR11, Thm. 1][Mau11, Thm. 3])**.** *The construction* $\to$ *is* (generally) composable*, i.e. for all* $(\varepsilon, \varepsilon') \in \mathbb{R}^+$*,* $(\mathcal{R}, \mathcal{S}, \mathcal{T}) \in \Phi^3$*,* $\pi \in \Sigma^2$*:*

- *we have sequential composability:* $(\mathcal{R} \xrightarrow[\varepsilon]{\pi} \mathcal{S} \wedge \mathcal{S} \xrightarrow[\varepsilon']{\pi'} \mathcal{T}) \Rightarrow \mathcal{R} \xrightarrow[\varepsilon+\varepsilon']{\pi \circ \pi'} \mathcal{T}$*,*
- *we have parallel composability:* $(\mathcal{R} \xrightarrow[\varepsilon]{\pi} \mathcal{S} \wedge \mathcal{R}' \xrightarrow[\varepsilon']{\pi'} \mathcal{S}') \Rightarrow \mathcal{R} \| \mathcal{R}' \xrightarrow[\varepsilon+\varepsilon']{\pi | \pi'} \mathcal{S} \| \mathcal{S}'$
- $\mathcal{R} \xrightarrow[0]{id} \mathcal{R}$

where $|$ *(resp.* $\circ$*) represents the parallel (respectively serial) composition of protocols,* $\|$ *is the merging of resources, and id is the identity converter.*

Chapter 3

Complexity Limitations of Classical Client Delegated Quantum Computing

One of the main questions that arises in secure delegated computations is whether it is possible to delegate a computation to a server in such a way that he learns nothing, in an *information-theoretic* sense, about the input of the computation.
More precisely, we address this question in the setting where the client is entirely classical and the server has a quantum computer. Therefore, we want to study whether the client can securely delegate any efficient quantum computation to the server. We emphasize here that as the client is entirely classical, this also implies that the communication between client and server must also be entirely classical. In the remaining of the chapter we will denote any protocol achieving information-theoretic secure classical delegation of quantum computations an an ITS-CDQC protocol.

In this chapter we focus on 2 main results: secure delegation of *decision* BQP problems and secure delegation of *sampling* BQP problems.
For the secure delegation of decision problems we show that if the client-server communication consists of $O(n^d)$ bits, then the existence of such an ITS-CDQC protocol would imply the following complexity theory result: $\mathsf{BQP} \subset \mathsf{MA}/O(n^d)$. We then give strong evidence that this complexity relation is unlikely by constructing an *oracle* O with respect to which this relation does not hold: $\mathsf{BQP} \not\subset \mathsf{MA}/O(n^d)$.
For our second result, we prove that if there exists a secure delegation protocol for quantum sampling problems (such as Boson Sampling), then there exist non-uniform circuits of size $2^{n-\Omega\left(\frac{n}{\log(n)}\right)}$, making polynomially queries to an $\mathsf{NP}^{\mathsf{NP}}$ that can compute the *permanent* of an $n \times n$ matrix, where n refers to the size of the client's input. While this result does not directly imply an impossibility of delegating sampling problems, it

reveals a connection between this task and the complexity of computing the permanent of a matrix. Even though the latter problem has been studied for a very long time, there are still very few algorithms (Ryser's algorithm from 1963 is still considered one of the most promising candidates) and very few results on the complexity analysis for the computation of a matrix permanent.

The problem of classical delegation from a classical client to an unbounded server has been studied for many decades. Our starting point represents the work of Abadi, Feigenbaum and Kilian who gave a characterization of what complexity classes of problems can be computed in this classical secure delegation setting, by first defining a framework that would encompass these families of protocols that achieve this task, framework called *Generalised Encryption Scheme (GES)*. In our work, in order to analyse the complexity theoretic implications of (information-theoretical) secure delegation of quantum computations, we use the GES framework introduced by Abadi et al [AFK87]. Essentially, a GES is a protocol between a probabilistic polynomial-time (BPP) classical Client and a computationally *unbounded* Server, in which the Server computes on encrypted data. The Client would like to compute some predicate $f : \{0,1\}^n \to \{0,1\}$, but as she is computationally bounded, she would like to achieve this by taking advantage of the power of the untrusted Server. In the GES, the Client would first send to the Server a description of f. Then she would encrypt her private input $x \in \{0,1\}^n$ using a BPP algorithm Enc and would send the result $Enc(x)$ to the Server. After this, the Client and the Server are allowed to interact for a number of rounds which is at most polynomial in n. After the interaction takes place, in the last stage of the GES, the Client uses a BPP algorithm Dec and applies to the entire transcript. As a result the Client obtains the desired outcome $f(x)$ with probability at least $\frac{1}{2} + \frac{1}{poly(n)}$.
But, very crucially, from a security point of view, after the GES protocol the Server should be able to learn nothing about the Client's input x, apart from its length n. As we considered the Server to be unbounded, the GES requires information-theoretic security. In the work of Abadi et al, they give a complexity theoretic characterization of all the predicates f that can be securely delegated in a GES protocol. More precisely, they showed that any predicate f that can be computed in a GES, while leaking to the Server only the size of the input must belong to the class $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. We show an alternative proof of this result in the next section. This result additionally, implies that no NP-Hard functions can be securely delegated in a GES scheme, by using the result that if NP-Hard problems would be included in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ then

the polynomial hierarchy would collapse.
Therefore, the two main questions that we address in this work can be rephrased as: can BQP computations be information-theoretically secure delegated using a GES scheme? Following the previously mentioned result this would be equivalent to studying the relation between the quantum class BQP and the classical complexity class NP/poly ∩ coNP/poly.
because we are both interested in delegating decision and sampling BQP problems, we study two variants of the GES framework: 1) delegating decision problems when the total communication between Client and Server is bounded by a polynomial $O(|x|^d)$, for an arbitrary but known constant d, and 2) delegating arbitrary sampling problems.
For the first result, knowing a bound on the (polynomial) size of the communication, we show that it implies that: BQP is a subset of $\mathsf{MA}/\mathsf{O}(\mathsf{n}^{\mathsf{d}})$. Then, we prove that this relation is unlikely by providing an oracle separation between these 2 classes.
For the second category, we prove that having a GES for BQP sampling problems (SampBQP) implies that there exist circuits for computing the permanent of a matrix more efficiently than it is believed to be possible. Regarding the first main result, it can be argued that oracle results do not represent the strongest evidence for relations between complexity classes. As an example, there exists oracles relative to which we both have that P and NP are equal and distinct. Nevertheless, using oracles we can study the query complexity of problems in different computational models. Furthermore, oracle results have also lead to important progress in complexity theory and developing new algorithms [GGH+13, Aar10].

3.1 Classical Delegation of Decision Problems

In this section we address the question of securely delegating BQP decision problems. As already discussed, Abadi et al showed that any function f that can be computed in a GES must satisfy:

Lemma 3.1.1. *[From [AFK87]] If f can be computed in a GES which leaks only the size of the input, then $f \in$ NP/poly ∩ coNP/poly.*

Additionally, they proved that if NP - Complete problems would be securely delegated using a GES protocol, then this would imply that NP - Complete $\subseteq$ NP/poly ∩ coNP/poly. This in turn would lead to a collapse of the polynomial hierarchy at the third level, as showed by Yap [Yap83]. Therefore, it seems highly unlikely that NP-

Complete problems can be information-theoretically secure delegated by a classical Client.

Then, to answer our question, whether BQP-Hard problems can be classically secure delegated, then the characterization of GES implies that this task would be possible only if any BQP-Hard problem can be computed in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

While ideally, we would like to have that similarly such a complexity relation would imply a collapse of the polynomial hierarchy, even the containment $\mathsf{BQP} \subseteq \mathsf{P}$ is not known to lead to a collapse of the PH.

To indicate an impossibility of delegating securely BQP computations, we instead consider a variant of GES where, we know in advance a bound on the size of the communication between Client and Server, in other words, considering their communication is upper bounded by a polynomial of fixed degree d in the size of the input. The rest of the GES remains exactly the same, with no restriction on the polynomial running time of the Client. As a consequence, we show that having such a GES for efficient quantum computations would imply: $\mathsf{BQP} \subseteq \mathsf{MA/O}(\mathsf{n^d}) \cap \mathsf{coMA/O}(\mathsf{n^d})$.

Then, to indicate that this relation is unlikely we show an oracle separation between the classes BQP and $\mathsf{MA/O}(\mathsf{n^d}) \cap \mathsf{coMA/O}(\mathsf{n^d})$:

Theorem 3.1.2. *For any $d \in \mathbb{N}$, there exists an oracle $\mathbb{O}_d$ such that:*

$$\mathsf{BQP}^{\mathbb{O}_d} \not\subseteq \mathsf{MA/O}(\mathsf{n^d})^{\mathbb{O}_d} \tag{3.1}$$

We remark that an interesting further question would be if the same separation between BQP and $\mathsf{MA/O}(\mathsf{n^d})$ can be achieved assuming an oracle that can be only accessed through classical queries by both the quantum and the classical algorithms. However, achieving this extension does not follow directly from our proof techniques and it would be non-trivial to deduce such an oracle separation.

Our result shows that relative to an oracle $\mathbb{O}_d$, there exist BQP problems that cannot be securely delegated by a classical Client in a GES with known bound on the communication.

We give a constructive proof for the family of oracles $\{\mathbb{O}_d\}_d$, which will be based on a version of the complement of Simon's problem [Sim97].

Specifically, for Simon's problem we have black-box access to a function $f : \{0,1\}^n \to \{0,1\}^n$, that is promised to be either a permutation or a 2-to-1 function: there exists a non-zero string s such that if $f(x) = f(y)$ for $x \neq y$ then $y = x \oplus s$. To solve the problem we need to decide which is the type of the function f.

The problem is known to be in BQP (relative to the function oracle) due to Simon's algorithm that requires only $O(n)$ queries to the function f [Sim97]. For the complement of this problem, called coSimon, in which we accept the input if the function f is 2-to-1, it can be shown that coSimon cannot be solved by an NP algorithm (and not even by a MA algorithm). Therefore, coSimon shows us an oracle separation (relative to the function f oracle) between the classes BQP and MA. Now, to use Simon's problem as a problem that cannot be solved by a GES, we would have to analyze its complexity relative to advice classes (such as NP/poly). As recalled from subsection 2.2.1, for advice classes the algorithm receives an additional advice string which is the same for all inputs of same size. However, for Simon's problem, for all inputs of size n, we have a unique underlying oracle function $f : \{0,1\}^n$. Therefore, Simon's problem (or its complement) can be solved by a polynomial time algorithm with a single bit of advice: for any input of size n the advice would be 0 if the function is a permutation and 1 otherwise. For this reason, we need to modify the structure of Simon's problem.
Essentially, to show our first main result, we consider that the black-box function is *input-dependent*. And to solve the problem, one needs as before, to determine whether the function specified by the input is a permutation or 2-to-1. In this case we can show by considering that our black-box functions are defined on a larger domain, $f : \{0,1\}^{n^D} \to \{0,1\}^{n^D}$, then the problem, called D-coSimon can still be solved in BQP, but very importantly it cannot be solved in $\mathsf{MA/O(n^d)}^{\mathbb{O}_d}$, for any constant d and a suitably chosen $D > d$ (as explained later). To show this separation, we will use a diagonalisation argument.
Unfortunately, we cannot use the same oracle to show the separation between BQP and NP/poly (or MA/poly) as D must depend on d and we would need to first know a bound on the size of the advice (the degree of the polynomial) in order to fix the size of the domain of our functions.

Next we describe formally the Generalised Encryption Scheme framework and we give an alternative proof for Lemma 3.1.1.

3.1.1 Generalised Encryption Scheme

A Generalised Encryption Scheme describes a 2-party protocol between an honest classical Client C and a malicious unbounded Server S, where the aim of C is to delegate a hard computation to S.
Formally, the components of the GES are the following:

- A predicate function $f : \{0,1\}^* \to \{0,1\}$;
- A plaintext input $x \in \{0,1\}^*$, for which C wants to learn the outcome $f(x)$;
- A BPP algorithm for key generation K, such that for any input $x \in \{0,1\}^*$, $K(x)$ outputs $(k, \text{success})$ with probability at least $\frac{1}{2} + \frac{1}{poly(|x|)}$, where $k \in \{0,1\}^{poly(|x|)}$. If the algorithm does not succeed, then it outputs (k', fail), where $k' \in \{0,1\}^{poly(|x|)}$;
- A polynomial-time deterministic algorithm E such that for any input $x \in \{0,1\}^*$, any key $k \in \{0,1\}^{poly(|x|)}$ and any message $s \in \{0,1\}^{poly(|x|)}$, $E(x,k,s)$ outputs an encryption $y \in \{0,1\}^{poly(|x|)}$;
- A polynomial-time deterministic algorithm D such that for any input $x \in \{0,1\}^*$, any key $k \in \{0,1\}^{poly(|x|)}$ and any message $s \in \{0,1\}^{poly(|x|)}$, $D(s,k,x)$ outputs a decryption $z \in \{0,1\}^{poly(|x|)}$

And they satisfy the following properties:

1. The rounds of communication between C and S are $m = poly(|x|)$. We will denote with c_i the message that C sends in round i, and with s_i the message sent by S at round i;
2. Given input x, C runs the key generation algorithm $K(x)$ until it succeeds - outputs $(k, \text{success})$. This step can be thought as preprocessing, taking place before the communication between C and S starts, and the same key k will be used in all the remaining stages of the protocol;
3. In round i, C will compute the encryption $c_i := E(x,k,\bar{s}_{i-1})$, where $\bar{s}_{i-1}$ represents the transcript of messages received by C from S until round i. Then C sends the encryption c_i to S;
4. In round i, after receiving c_i, S will respond with a message s_i;
5. After the communication between C and S ends, as a last step C uses algorithm D to compute $z := D(\bar{s}_m,k,x)$. With probability at least $\frac{1}{2} + \frac{1}{poly(|x|)}$, z is equal to the target computation $f(x)$.

Roughly speaking, the purpose of a GES scheme is to allow a computationally bounded Client to obtain an outcome $f(x)$ for an input of her own choice x, which she cannot compute using her limited computational resources. To obtain this desired computation she interacts with a computationally unbounded, but malicious Server, for a number of

rounds which is polynomial in the length of the input.
But very importantly, the GES also ensures security against the malicious Server, more precisely the *privacy* for the Client's input. As the Server is unbounded, the privacy must hold an information theoretic sense, and we define it formally in the following manner:

Definition 3.1.3. *Let X be the random variable denoting the input of C in a GES, $T(X)$ be the random variable for the transcript during the protocol running on input X and $L(X)$ a function of X, which we will call leakage. We say that the GES protocol leaks at most $L(X)$ if and only if:*
X and $T(X)$ are independent given $L(X)$.

Before showing the alternative proof of Lemma 3.1.1, we must first introduce the notion of *randomized advice*. Apart from the standard advice classes introduced in subsection 2.2.1, where the advice is a fixed deterministic function of the size of the input, we also introduce here an advice complexity class, where the advice is *randomized*, meaning that for each input length n, the advice is sampled according to some distribution $\mathcal{D}_n$ (distribution parametrized by size of input). A very important relation regarding randomized advice class, which we will also use in our work, is the following:

Lemma 3.1.4. *[From [Aar06]]* $\mathsf{MA/rpoly} = \mathsf{MA/poly} = \mathsf{NP/poly}$

Now, we can present the simplified proof of Lemma 3.1.1, by first showing that any function computable in a GES belongs to the class $\mathsf{MA/rpoly}$.

Lemma 3.1.5. *If a function f admits a GES leaking at most the size of the input, namely $L(X) = |X|$, then $f \in \mathsf{MA/rpoly}$.*

Proof. Consider a predicate f which admits a GES leaking at most the size of the input.
To begin with, we consider a simplified GES, in which the communication consists of a single round between Client C and Server S. As a result, the GES protocol running on input x can be described in the following steps:

1. C runs $K(x)$ until it successfully outputs an encryption key k;

2. C computes the encryption $c := E(x,k,'')$ and sends y to S;

3. C receives a response s from S;

4. C runs the decryption algorithm $D(s,k,x)$ and obtains z, such that with probability $\frac{1}{2}+\frac{1}{poly(|x|)}$ satisfies: $z=f(x)$.

Based on this GES that leaks at most the size of x, we will construct a MA/rpoly algorithm for f. In other words, we will give a probabilistic polynomial-time algorithm that receives a checkable witness w and a randomized polynomial-sized advice (sampled from a distribution which is the same for inputs of equal length) which can compute $f(x)$.
The algorithm is described in the following way.
Consider an input x of length n. Then the randomized advice r for all inputs of length n is constructed in the following way:

1. We sample at random x_n in $\{0,1\}^n$;

2. We compute a successful key in a GES for input x_n: $k_n := E(x_n)$;

3. We compute the encryption: $y_n := E(x_n,k_n,'')$ and send it to S;

4. We receive s_n as response from S;

5. The advice is $r_n=(x_n,k_n,y_n,s_n)$.

We consider two cases. If $x_n=x$, then the MA algorithm uses the decryption algorithm D on input (r_n,k_n,x_n) and will obtain $f(x)$ with probability $\frac{1}{2}+\frac{1}{poly(n)}$ and we are done. Now consider $x_n \neq x$. From the security condition, we know that the GES leaks about x at most n. This implies that there must exist some key k such that $y_n = E(x,k,'')$. To see this, suppose by contradiction that for any possible key k we would have $E(x,k,'') \neq y_n$. Then if in the GES described above, S receives from C the message y_n and as a result he knows that the Client's input cannot be equal to x, and therefore he learned more about the input of C, than he is allowed to (the size of input). More formally, this would imply that the input and the transcript of the protocol are not independent given the size of the input, as some certain transcripts (containing y_n) can only take place for certain input (different than x).
Now that we made sure that there exists a key k such that $y_n = E(x,k,'')$, we will use k as the witness for the MA algorithm. Namely, the polynomial-time verifier given the witness k will check that the y_n from the advice verifies the equality: $y_n = E(x,k,'')$.
Finally, the algorithm will run the GES decryption algorithm D, $D(r_n,k,x)$ and from the

correctness of the GES, the result would be equal to $f(x)$ with probability $\frac{1}{2} + \frac{1}{poly(n)}$. This shows the MA algorithm receiving a randomized advice that computes $f(x)$ which concludes our proof.

Therefore, $f \in \mathsf{MA/rpoly}$ and using Lemma 3.1.4, we also have that $f \in \mathsf{NP/poly}$.

Furthermore, as the key k must exist independent of the value of the predicate $f(x)$, then the algorithm we presented above holds for both cases $f(x) = 0$ and $f(x) = 1$ and hence the same algorithm works both when x is a yes instance and a no instance. As a result, we also have that $f \in \mathsf{coNP/poly}$ and consequently $f \in \mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

To finish our proof, we need to show that this result holds also in the case when the GES consists of multiple rounds of communication. For the general case, we follow the previous strategy which generalizes in a straightforward way.

From the privacy condition of the GES, we know that any transcript consisting of polynomially in n number of rounds can depend only on n.

Therefore, we will make the advice for inputs of length n to be the entire transcript of a GES protocol, drawn from the distributions of possible GES transcripts for inputs of length n. This distribution will correspond to the parametrized distribution corresponding to the randomized advice.

Similarly as before, the witness the MA algorithm will receive is a key k which will make the input x compatible with the transcript from the advice. And from the security condition, we know such a key k must exist. Finally, the MA procedure will run the GES decryption for k, x and the advice transcript and from the correctness condition will obtain the desired outcome $f(x)$ with probability $\frac{1}{2} + \frac{1}{poly(n)}$. □

Interestingly, we observe that if the communication between C and S consists of $O(n^d)$ messages, for some constant d, then we can also characterize to which complexity class f belongs. Namely, following the above proof we notice that the advice is exactly a GES transcript, therefore $O(n^d)$ bits of advice would suffice to compute f. As a result we have that:

Corollary 3.1.6. *If a function f admits a GES with communication bounded by $O(n^d)$, then $f \in \mathsf{MA/O(n^d)}$.*

In the next section we present our main result about the delegation of BQP decision problems in a GES with a known bound on the communication. To give evidence that BQP computations cannot be information-theoretically secure delegated in such a GES, we will show an oracle separation between the classes BQP and $\mathsf{MA/O(n^d)}$, as stated in Theorem 3.1.2.

3.1.2 Oracle separation between BQP and MA/O(n^d)

In order to show the separation between the classes BQP and MA/O(n^d) we will construct an oracle based on the complement of Simon's problem.

We recall that for Simon's problem, the input is a function $f : \{0,1\}^n \to \{0,1\}^n$, to which we only have black-box access and such that f is promised to be either 1-to-1 (permutation) or 2-to-1 (there exists $s \in \{0,1\}^n - \{0\}$, such that for any $x \neq y$, $f(x) = f(y)$ if and only if $y = x \oplus s$). To solve the problem we need to determine of which type f is. Namely, for Simon's problem the algorithm must accept if f is 2-to-1 and reject if it is 1-to-1. And thus, for the complement of Simon's problem, an algorithm must accept if f is 1-to-1 and reject otherwise.

As mentioned, for this problem we are not given an actual description of the underlying function, but only oracle access to f. In this way, Simon's problem offers an oracle separation between the classes BPP and BQP, as a classical algorithm (with oracle access to f), requires $O(2^{\frac{n}{2}})$ queries to f, whereas a quantum algorithm only needs $O(n)$ queries to f.

More precisely, the oracle is a function $\mathbb{O} : \{0,1\}^* \to \{0,1\}^*$ such that for any $n \in \mathbb{N}$, if we consider the restriction of $\mathbb{O}$ to the domain $\{0,1\}^n$, and we denote it by $\mathbb{O}_n : \{0,1\}^n \to \{0,1\}^n$, then $\mathbb{O}_n$ corresponds to either a 1-to-1 or a 2-to-1 function.

As a result, we can naturally define a language which lies in $\mathsf{BQP}^{\mathbb{O}}$, but is not contained in $\mathsf{BPP}^{\mathbb{O}}$:

$$L(\mathbb{O}) = \{0^n \,|\, \mathbb{O}_n \text{ corresponds to a 2-to-1 function}\} \tag{3.2}$$

Even more powerful, we can use the complement of this language, defined as: $L^c(\mathbb{O}) = \{0^n \,|\, \mathbb{O}_n \text{ corresponds to a 1-to-1 function}\}$ in order to show a separation between BQP and NP with respect to the oracle $\mathbb{O}$ [Aar10]. Next we will also show a proof for this separation for a modified version of the oracle, which will then help us construct our main oracle for the separation between BQP and MA/O(n^d).

A GES for securely delegating BQP problems would imply that $\mathsf{BQP} \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Therefore, ideally we would aim to show that there exists an oracle $\mathbb{O}$ for which $\mathsf{BQP}^{\mathbb{O}} \not\subseteq \mathsf{NP/poly}^{\mathbb{O}}$.

And the intuition behind constructing this oracle $\mathbb{O}$ would be the following: instead of considering a single function $\mathbb{O}_n$ for each possible input length n, we would construct instead a different oracle function $\mathbb{O}_x$ for any input $x \in \{0,1\}^n$. As a result for an input length n, we would have 2^n functions that need to be decided whether they are 1-to-1 or 2-to-1.

However a classical *NP* algorithm with oracle access to $\mathbb{O}$ will receive only a polynomial (in n) bits of advice, which will be the same advice string for all these 2^n functions. And as a result, it seems that this amount of advice would be insufficient to make the $\mathsf{NP}^{\mathbb{O}}$ algorithm decide correctly for all these inputs. However, we will see later that formalizing this intuition for any polynomial advice is problematic.
For this reason, we consider that we know in advance a bound on the degree of the polynomial advice. In other words, assuming that this degree is d, then we can construct an oracle showing our oracle separation: $\mathsf{BQP}^{\mathbb{O}} \not\subseteq (\mathsf{MA}/\mathsf{O}(\mathsf{n}^{\mathsf{d}}))^{\mathbb{O}}$.
To construct this oracle, we will show an incremental proof, starting from an oracle separation between the classes BQP and NP. While separation of these 2 classes with respect to an oracle are already known, including using the complement of Simon's problem, in our case we will show this separation for a different variant of Simon's problem, where instead of assigning a different function to each input size, we will assign a different function for each different input.

Lemma 3.1.7. *There exists an oracle $\mathbb{O}$ based on the complement of Simon's problem such that:*

$$\mathsf{BQP}^{\mathbb{O}} \not\subseteq \mathsf{NP}^{\mathbb{O}} \tag{3.3}$$

Proof. We will define the oracle $\mathbb{O}$, together with a language based on $\mathbb{O}$, $coSimon(\mathbb{O})$ such that:

$$coSimon(\mathbb{O}) \in \mathsf{BQP}^{\mathbb{O}} \text{ and } coSimon(\mathbb{O}) \notin \mathsf{NP}^{\mathbb{O}} \tag{3.4}$$

Specifically, for each n we will consider a family of 2^n functions:
$\mathcal{F}^n = \{f_i^{(n)} : \{0,1\}^n \to \{0,1\}^n\}_{i\in\{0,1\}^n}$.
Then, we define the oracle $\mathbb{O}$ on input 1^n, $i \in \{0,1\}^n$ and $x \in \{0,1\}^n$ as:

$$\begin{aligned} &\mathbb{O}(1^n, i, x) := f_i^{(n)}(x) \\ &\mathbb{O}(1^n, i) \text{ corresponds to the function } f_i^{(n)} \end{aligned} \tag{3.5}$$

Now, we define the language $coSimon(\mathbb{O})$ as:

$$coSimon(\mathbb{O}) = \{(1^n, i) \,|\, f_i^{(n)} \text{ is a 1-to-1 function } \} \tag{3.6}$$

The problem we are constructing is a promise problem and the language $coSimon(\mathbb{O})$ is the set of "yes" instances of the problem, whereas the set of "no" instances is not the complement of this set, but is the language: $coSimon^c(\mathbb{O}) = \{(1^n, i) \,|\, f_i^{(n)} \text{ is a 2-to-1 function } \}$.
From this definition we can see that the oracle $\mathbb{O}$ gives access to the functions for which

we need to tell whether they are 1-to-1 or 2-to-1. The main point is that for this problem, as for the standard Simon's problem, the algorithms only have black-box access to these functions and the only way they can solve this problem is by querying the oracle $\mathbb{O}$.

We also know that given $\mathbb{O}$, the language $coSimon(\mathbb{O})$ belongs to the class $\mathsf{BQP}^{\mathbb{O}}$, as we can just run Simon's algorithm on an input $(1^n, i)$ and then flip the acceptance and rejection answers.

As standard in quantum query complexity, querying the quantum oracle $\mathbb{O}$ is realised through the following unitary operation, and has the following form:

$$|1^n\rangle |i\rangle |x\rangle |y\rangle \xrightarrow{\mathbb{O}} |1^n\rangle |i\rangle |x\rangle |y \oplus \mathbb{O}(1^n, i, x)\rangle \tag{3.7}$$

Therefore, $\mathbb{O}$ on input $(1^n, i, x)$ will output the value of the function $f_i^{(n)} : \{0,1\}^n \to \{0,1\}^n$ evaluated on the point x, where $f_i^{(n)}$ is either 1-to-1 or 2-to-1 . We can see that the size of the input tuple is $3n$ as the first input specifying the domain size is given in unary and the index of the function and the element on which we want to evaluate the function are given in binary. Additionally, the oracle is defined for all n and for any $i, x \in \{0,1\}^n$.

Now that we defined abstractly how the oracle $\mathbb{O}$ works, we will now show how to construct the adversarial oracle $\mathbb{O}$, and more specifically, how to construct the family of functions $\mathcal{F}^n$, for any n. By adversarial we mean that we will define $\mathcal{F}^n$ in such a way that every non-deterministic algorithm using the oracle $\mathbb{O}$ will fail to decide correctly the language $coSimon(\mathbb{O})$. The following proof will use a diagonalisation argument.

Using the fact that the set of NP Turing Machines is countable, we consider an enumeration of them and pick the k-th NP machine, denoted by M_k. Now we will study its behaviour for input size $n = k + n_0$, for some $n_0 > 0$ defined later on.

Now suppose some function index $i \in \{0,1\}^n$, and we fix a 1-to-1 function as $f_i^{(n)}$.

By simulating the behaviour of M_k on the input $(1^n, i)$, we can check to see whether M_k accepts or not $f_i^{(n)}$. Recall that for $coSimon(\mathbb{O})$ accepting means the input function is 1-to-1 and rejecting means the input function is 2-to-1 .

If M_k rejects the input $(1^n, i)$, then M_k gave the wrong answer and we are done. On the other hand, if M_k accepted the input, then from the definition of NP TM, we know that there exist a polynomial-sized path in the non-deterministic computation tree of M_k, which ends in an acceptance state. Let us denote this acceptance path with π and its length with $l' = poly(n)$.

Then we know that M_k can make at most l' queries to the function oracle $\mathbb{O}$, and we will denote the list of $l < l'$ queries that M_k will make and the answers from $\mathbb{O}$ to these queries as:

$$Q = [(x_1, f_i(x_1)), (x_2, f_i(x_2)), \cdots, (x_l, f_i(x_l))] \tag{3.8}$$

where $x_1, \cdots x_l \in \{0,1\}^n$ are the queried elements, chosen by M_k, during the path π, as depicted in Figure 3.1.

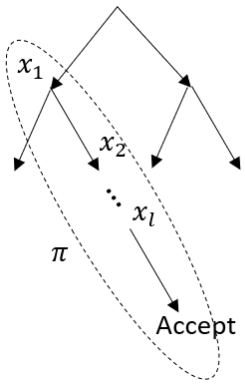


Figure 3.1: Computational tree of M_k and the queries to $\mathbb{O}$ during the accepting path π

We now construct a new function g_i, which will be a 2-to-1 function such that g_i will have the same values as f_i in the queried points $x_1, x_2, \cdots, x_l$, namely: $g_i(x_1) = f_i(x_1), \cdots, g_i(x_l) = f_i(x_l)$.

But first we need to show that we construct such a 2-to-1 function g_i. Since g_i is 2-to-1 , then there exists a non-zero string called xor-mask s such that $g_i(x) = g_i(y)$ if and only if $x = y$ or $y = x \oplus s$. The number of xor-masks s is $2^n - 1$.

Now as g_i is equal to f_i in l different points and as f is a permutation, then it must be that g_i also produces different images for each of those $l = poly(n)$ inputs. This implies that we need to choose the xor mask s for g_i different than the xor of any 2 of those points: $\forall 1 \leq i, j \leq l$, $i \neq j$, we have that $s \neq x_i \oplus x_j$. This means that we need to eliminate $\binom{l}{2}$ values for s. But the total number of possible s is $2^n - 1$ and as $l = poly(n)$, there exist sufficiently large n such that $2^n - 1 \geq \frac{l(l-1)}{2} = poly(n)$.

Therefore, we choose n_0 such that $2^n - 1 > poly(n)$ and as there exist such xor masks s satisfying the above condition and we just need to pick one of them to construct the 2-to-1 function g_i.

Finally, for the oracle $\mathbb{O}$, we change the function at index i, by replacing the 1-to-1 f_i with the 2-to-1 g_i. In this case, we are guaranteed that the path π of the computational tree of M_k will remain an accepting path, and therefore M_k will decide incorrectly that the input $(1^n, i)$ corresponds to a 1-to-1 function.

Then using this construction, for any non-deterministic TM, we will construct an input for which the machine will decide the language $coSimon(\mathbb{O})$ incorrectly, hence:

$$coSimon(\mathbb{O}) \notin \mathsf{NP}^{\mathbb{O}} \tag{3.9}$$

which concludes the proof.

□

Next, we can show a stronger oracle separation, between BQP and MA. Namely, by using the proof technique from Lemma 3.1.7, we can show that even if the NP algorithm receives additionally a polynomial amount of randomness, it cannot decide correctly the language $coSimon(\mathbb{O})$. As a result $coSimon(\mathbb{O})$ lies outside the class MA relative the oracle $\mathbb{O}$.

Lemma 3.1.8. *There exists an oracle* $\mathbb{O}$ *such that* $\mathsf{BQP}^{\mathbb{O}} \not\subseteq \mathsf{MA}^{\mathbb{O}}$.

Proof. As opposed to the previous case, an MA TM is an NP machine that can also use randomness and essentially can be thought as a probability distributions over NP algorithms.
The main idea for this proof will be to pick an oracle at random and then reduce the problem to the NP case. Suppose that the function oracle is 1-to-1 or 2-to-1 with equal probability and that in both cases the specific function is chosen uniformly at random. If the complement of Simon's problem with respect to the random oracle lies in MA, this implies that there must exist an NP algorithm that can decide the problem correctly with probability at least $\frac{2}{3}$ over the choice of the oracle function.
Now, we proceed with a proof by contradiction. Suppose there exists such a NP TM M_k.
Then we pick an input which is accepted by M_k, and we denote it by $(1^n, i)$. As M_k accepts $(1^n, i)$, then there must exist an accepting path in the computational tree of M_k along which the machine makes $l = poly(n)$ queries to the oracle function.
If the underlying function $f_i^{(n)}$ is 2-to-1 (with xor-mask s), then from the proof of Lemma 3.1.7, we know that the probability of distinguishing $f_i^{(n)}$ from a 1-to-1 function is given by the probability of finding a collision after l queries, which is bounded by:

$$\begin{aligned} &\Pr[collision] \leq \Pr\left[\exists j \in \{1,\cdots,l-1\} \,|\, f_i^{(n)}(x_l) = f_i^{(n)}(x_j)\right] \\ &= \Pr\left[\exists j \in \{1,\cdots,l-1\} \,|\, s = x_l \oplus x_j\right] = \frac{l-1}{2^n - 1 - \frac{l(l-1)}{2}} \end{aligned} \tag{3.10}$$

But $l = poly(n)$, therefore $\Pr[collision]$ is exponentially small in n.
Then, as for any input (including (i,n)) the probability that the underlying function $(f_i^{(n)})$ is 1-to-1 or 2-to-1 is $\frac{1}{2}$, then we obtain that the probability that M_k accepts correctly the input (i,n) is at most: $\frac{1}{2} + 2^{-\Omega n} < \frac{2}{3}$ (for sufficiently large n). This ends our contradiction proof.

□

In the next part we move to classical advice classes. Namely, we first show an oracle separation between the quantum polynomial time TM and the classical polynomial time machines that receive a bounded polynomial of advice. The proof techniques required for this result, will also help us show our main oracle separation between BQP and $\mathsf{MA/O(n^d)}$.

Lemma 3.1.9. *For any $d \in \mathbb{N}$, there exist an oracle $\mathbb{O}$ such that:* $\mathsf{BQP}^{\mathbb{O}} \not\subseteq \mathsf{P/O(n^d)}^{\mathbb{O}}$.

Proof. The class $\mathsf{P/O(n^d)}$ refers to the class of problems which can be solved by a deterministic polynomial-time TM which receives an additional advice of size $O(n^d)$, where d is a constant and n specifies the size of the input (as we have seen, in our case the input size is $2n$, as n represents the size of the inputs for our oracle functions).
As opposed to the previous analysis, instead of having the ability to non-deterministically choose one of exponentially many paths, we will deal with polynomial-time deterministic algorithm M that will receive some non-uniform information in order to decide the language $coSimon(\mathbb{O})$. Essentially, each possible advice will determine a new behaviour for M, which can even involve a different sequence of queries to the oracle depending on the advice.
What we want to prove is that irrespective of what advice M will receive, M cannot decide correctly $coSimon(\mathbb{O})$ for all inputs. To show this, we first need to consider functions over a larger domain than n-bit strings.
More specifically, for any $d > 0$, we will choose $D > d$ such that the family of functions $\mathcal{F}^n$ will contain 2^n 1-to-1 or 2-to-1 functions f defined on the domains:
$f : \{0,1\}^{n^D} \to \{0,1\}^{n^D}$. Then for any d, the oracle denoted as $\mathbb{O}_d$ will receive queries of the form:
$(1^n, i, x)$ - where $i \in \{0,1\}^n$ is the index of the function within $\mathcal{F}^n$ and $x \in \{0,1\}^{n^D}$ is an element from its domain.
In order to show that this oracle can be used to prove the separation, we first need to argue that this modified problem can still be solved in $\mathsf{BQP}^{\mathbb{O}_d}$.
This is indeed the case, since by expanding the domains of the functions we just change

the running time of Simon's algorithm from $O(n)$ to $O(n^D)$. But, as D is just a fixed constant, then the algorithm still runs in polynomial time and therefore $coSimon(\mathbb{O}_d) \in \mathsf{BQP}^{\mathbb{O}_d}$. What remains to be shown is that we also have: $coSimon(\mathbb{O}_d) \notin \mathsf{P}/\mathsf{O}(\mathsf{n}^{\mathsf{d}})^{\mathbb{O}_d}$.
As in the previous oracle separations, we will prove this by diagonalisation, by considering an enumeration of the deterministic polynomial time machines and we will show that no matter what advice the k-th polynomial time machine M_k receives, it cannot correctly decide $coSimon(\mathbb{O}_d)$.
However, the main challenge comes from the fact that each advice can induce a different behaviour to M_k and therefore we need to construct the oracle in such a way that all advice strings would lead to M_k failing to answer correctly for at least one input. This is contrast to the previous case where we only had to analyze the behaviour of one accepting paths on the non-deterministic computation tree.
Let us consider the behaviour of M_k for an input of size $n = k + n_0$, where the constant n_0 will be specified later.
As the advice is a string of size $O(n^d)$, there are $2^{O(n^d)}$ possible advice strings. Crucially, whichever of these strings M_k will use, it will be the same for for all 2^n inputs of length n (all 2^n functions from $\mathcal{F}^n$).
In order to construct the oracle for inputs of size n, we essentially need to construct the family of functions $\mathcal{F}^n$.
Hence, let us consider the first index inside $\mathcal{F}^n$, namely 0^n and assign to this index any 1-to-1 function $f : \{0,1\}^{n^D} \to \{0,1\}^{n^D}$. We will now inspect the behaviour of M_k for the index 0^n and for each possible advice string.
If for more than half of all the advice strings M_k rejects f, then we keep f in its current form for the index 0^n. This is because in this case, we have that half of the advice strings have been eliminated as they all lead to M_k giving a wrong answer.
If on the other hand, more than half of all the advice strings lead to M_k accepting f, we will attempt to turn f into a 2-to-1 function while keeping acceptance for those advice strings. As a result this would reduce to the previous case, and we would be able to again eliminate half of the advice strings.
For each advice, denoted by a_j, for $j \in \{0,1\}^{O(n^d)}$, the machine M_k will make a sequence of polynomially in n many queries to f.
We denote these sequences of queries together with their responses from the oracle as:

$$\sigma_j = [(x_1^j, f(x_1^j)), (x_2^j, f(x_2^j)), \cdots, (x_{l_j}^j, f(x_{l_j}^j))] \tag{3.11}$$

where each l_j is polynomial in n.

We now consider a 2-to-1 function $g : \{0,1\}^{n^D} \to \{0,1\}^{n^D}$ such that:
For all j such that M_k with advice a_j and querying σ_j, accepts the function f:

$$g(x_1^j) = f(x_1^j), g(x_2^j) = f(x_2^j), \cdots, g(x_{l_j}^j) = f(x_{l_j}^j) \tag{3.12}$$

In other words, if we place g on index 0^n, then we will get identical responses to the queries which make M_k accept this input. Since the number of queries inputs is $l_j = poly(n)$ and the number of advice strings is $2^{O(n^d)}$, then the maximum number of different inputs which can be queried is $l_j \cdot 2^{O(n^d)}$, which is also of order $2^{O(n^d)}$.
However, unlike the proofs for our previous results, this number of queried inputs is exponential in the size of the input of M_k, so we first have to make sure that such a 2-to-1 function even exists.
The trick is that we chose the domain of our functions through the variable D, and we can make it sufficiently large to accommodate for a 2-to-1 function satisfying the $2^{O(n^d)}$ restrictions mentioned above.
As before, because f is a permutation then no two queried values will have the same image. And therefore in order to construct the xor-mask $s \in \{0,1\}^{n^D}$ for the function g, we need to ensure that it is different from the xor of any 2 queried preimages. These will be the restricted values of the xor-mask s. The total number of pairs of queried inputs is at most $\binom{2^{O(n^d)}}{2} = \frac{2^{O(2n^d)} - 2^{O(n^d)}}{2}$, which is also of order $2^{O(n^d)}$. But the total number of possible xor masks is 2^{n^D}. Therefore, to make sure that such a xor mask s exists and hence such a 2-to-1 function g exists, it suffice to ensure that: $2^{n^D} > 2^{O(n^d)}$.
As a result if we set $D > d$, then we have a 2-to-1 function g that matches the responses of f for all $2^{O(n^d)}$ possible queries.
Therefore, by placing g on the index 0^n, we can eliminate half of the possible advice strings. Then, no matter how M_k behaves, for the input 0^n, we can eliminate half of the advices.
And now, we can repeat this procedure for the other inputs (function indices). We are effectively halving the number of possible advice strings with each index. Since in $\mathcal{F}^n$ there are exactly 2^n functions, in order to eliminate all possible advice strings, we need to ensure that: $\frac{2^{O(n^d)}}{2^{2^n}} < 1$, or equivalently, $O(n^d) < 2^n$. Then, to achieve this, we just have to choose n_0 (where $n = n_0 + k$) large enough such that this inequality holds.
Finally, we have that for any k and for any advice that M_k can receive, there always exists at least one input to $coSimon(\mathbb{O}_d)$ that is decided incorrectly by M_k, and as a result:

$$coSimon(\mathbb{O}_d) \notin \mathsf{P}/\mathsf{O}(\mathsf{n}^{\mathsf{d}})^{\mathbb{O}_d} \tag{3.13}$$

which concludes our proof. □

Remark: Note that the same proof technique cannot be used to show an oracle separation between BQP and P/poly. The reason is that the trick in our proof was to consider D - determining the size of the function domain to be larger than d - determining the size of the advice. This is clearly possible only when we know in advance a bound on the degree of the advice polynomial. If the advice size could be any arbitrary polynomial, then no matter how we would choose the oracle domain size parameter D, there would always exist some degree $d' > D$ of the advice for which our proof technique would not work.
A possible way to fix this issue, would to make D part of the input in some way such that it can increase as well. Hence, if D was included in the input as unary string $h(n)$, where h is an increasing function, then for sufficiently large n, we will have $g(n) > d$. But we can immediately notice the issue with this strategy. While through this construction indeed the problem cannot decided by a P/poly algorithm, it is also not solvable anymore in BQP. This is because in this case, the quantum query complexity required to run Simon's algorithm would become $O(n^{g(n)}$, which is no longer polynomial, as g is an increasing function, and cannot be constant. As a result, proving an oracle separation between BQP and P/poly seems to require some non-trivial modifications of our proof or a very different approach.

Finally, we are able to show our main result concerning the delegation of BQP computations in a GES with a known bound on the communication.

Theorem 3.1.10. *For each $d \in \mathbb{N}$, there exists an oracle $\mathbb{O}_d$ such that* $\mathsf{BQP}^{\mathbb{O}_d} \not\subseteq \mathsf{MA}/\mathsf{O}(\mathsf{n}^{\mathsf{d}})^{\mathbb{O}_d}$.

Proof. To begin with, we first show that there exists an oracle $\mathbb{O}_d$ relative to which the complement of Simon's problem does not belong to the class $\mathsf{NP}/\mathsf{O}(\mathsf{n}^{\mathsf{d}})$.
The oracle $\mathbb{O}_d$ will be constructed in the same way as we did for the $\mathsf{P}/\mathsf{O}(\mathsf{n}^{\mathsf{d}})$ case. The same proof technique can be applied here. Namely, we consider the k-th non-deterministic polynomial-time machine M_k and we will examine its behaviour for some input $(1^n, i)$, where $n = k + n_0$, with n_0 chosen as in Lemma 3.1.9.
For which index we initially choose a permutation and examine what M_k will do given any possible advice of length $O(n^d)$.
If more than half of the advice strings lead to M_k rejecting the input, then we keep this permutation for this input and proceed to the next one. Otherwise we will replace the permutation with a 2-to-1 function constructed in the following way. For each

advice for which M_k accepts, there must exist a path in the computational tree of size polynomial in n which leads to an accept node. We will pick one such accepting paths for each advice leading M_k to an accept. Then when constructing the 2-to-1 function we want to ensure that for every input queries along all these accepted paths, the 2-to-1 function will have the same images as the initial permutation function. This reduces the problem to the proof of Lemma 3.1.9. As a result, we know that by picking $D > d$, such a 2-to-1 function exists and therefore as before for each function index we are able to eliminate half of the possible advice strings. Finally, by choosing n_0 large enough to ensure that all advice will be eliminate after we process all inputs (after 2^n steps), we consequently have that the language $coSimon(\mathbb{O}_d)$ is decided incorrectly by all non-deterministic polynomial-time algorithms that receive an extra advice of size $O(n^d)$.

For the $\mathsf{MA/O(n^d)}$ case, we can use the same proof technique as in Lemma 3.1.8 to reduce the MA case to the NP setting. We therefore conclude that:

$$coSimon(\mathbb{O}_d) \not\subseteq \mathsf{MA/O(n^d)}^{\mathbb{O}_d} \tag{3.14}$$

□

3.2 Classical Delegation of Sampling Problems

We now want to study what would happen if we can securely and classically delegate a *sampling problem*, such as Boson Sampling to a quantum Server using a GES protocol. Boson Sampling defined by Aaronson and Arkhipov [AA11] is the problem of simulating the statistics of photons (bosons) passing through a linear optics network. In this problem we have an initial configuration of identical photons placed in known locations (referred to as modes). The photons then pass through the linear optics network, consisting of optical elements. Finally, we perform a measurement to determine the new location of photons, in the output modes of the system. To illustrate this process, we can think of the following example: imagine identical balls which are dropped sequentially from different initial locations, through a board containing pegs arranged as depicted in Figure 3.2. Then, the input of the problem is the arrangement of the pegs and the output is represented by the number of balls in each output location.

The main reason why this is referred to as a sampling problem is the fact that we have defined a probability distribution over the different configurations of photons in the output modes.

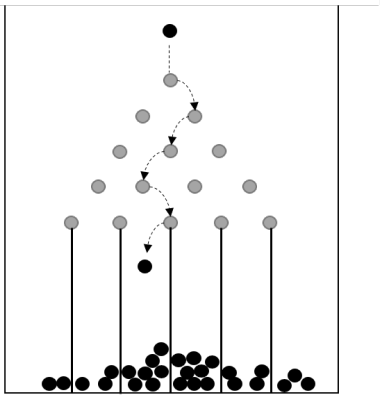


Figure 3.2: Analogy of the Boson Sampling problem

In the exact sampling version of this problem, called Exact Boson Sampling, which is the problem we will be considering for the GES delegation, the task is to output a sample from this specific probability distribution.

Regarding the complexity of Boson Sampling, Aaronson and Arkhipov [AA11] showed that the probability of observing a particular configuration of photons in this experiment is proportional to the squared permanent of a matrix describing the optical network. Moreover, they showed a separation between SampBQP and SampBPP by proving that no polynomial-time probabilistic algorithm can sample from this distribution, unless the polynomial hierarchy collapses at the third level.

As a result, while a quantum computer can simulate the optical network and sample from the underlying target distribution in polynomial time in the size of the network, it is highly unlikely that classical computers can achieve this.

Now, in the context of delegating Exact Boson Sampling, in a GES the classical Client's input would be a description of the linear optics network. The Client's target is to obtain after the interaction with the Server a sample from the Boson Sampling distribution associated with the input network, but such that the description of this network remains information-theoretically hidden from the Server, with the only leaked information to the Server being the size of the network.

Our main result about the delegation of Exact Boson Sampling can be summarized as follows:

If there exists a GES for exact Boson Sampling, then for any square $n \times n$ matrix X with elements in $\{-1,0,1\}$ there exists circuits C of size $2^{n-\Omega\left(\frac{n}{\log n}\right)}$ making polynomially-sized queries to an $\mathsf{NP}^{\mathsf{NP}}$ oracle for computing the permanent of the matrix X.

Regarding the hardness of computing the permanent of a matrix, this problem is known

to be #P-Hard. By Toda's theorem [Tod91], we know that this implies that if solving this problem would be possible at any level of the polynomial hierarchy, then the hierarchy would collapse at that level. Furthermore, regarding the exact complexity, the best known algorithm for computing the permanent of a matrix, developed by Björklund [Bjö16] has a running time of $2^{n-\Omega\left(\sqrt{\frac{n}{\log n}}\right)}$. Prior to this, the leading algorithm for computing the permanent of a matrix was Ryser's algorithm [Rys63], developed over 50 years ago, which requires $O(n2^n)$ arithmetic operations.
For these reasons, we conjecture that the circuits C cannot exist, and as a result, the Boson Sampling cannot be classically and securely delegated.

3.2.1 The Boson Sampling Problem

We recall from section 2.2 that for sampling problems, the input describes a probability distribution and the output is a sample from this distribution either exactly or approximately (sample from a close distribution).
In this work we will be studying exact sampling and more specifically the Exact Boson Sampling problem. In this problem identical bosons are sent through a linear optics network and then non-adaptive quantum measurements are performed in order to count the number of bosons in each output mode (output location).
In more details, if our quantum system has n bosons and m output modes, then any computational basis state of this system has the form: $|S\rangle = |s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_m\rangle$, where for any $i \in \{1,\cdots,m\}$, $s_i \in \{0,1,\cdots,n\}$ denotes the number of bosons in mode i. Hence we must have $s_1 + \cdots + s_m = n$. We denote by S, the tuple $(s_1,\cdots,s_m)$ such that $s_1 + \cdots + s_m = n$ and with $\mathbb{S}_{m,n}$ the set of all such tuples S. We notice that the cardinality of $\mathbb{S}_{m,n}$ is $|\mathbb{S}_{m,n}| = \binom{m+n-1}{n}$.
Using this computational basis, then the general quantum state of the system can be expressed as:

$$|\psi\rangle = \sum_{S \in \mathbb{S}_{m,n}} a_S |S\rangle \tag{3.15}$$

where $a_S \in \mathbb{C}$ such that: $\sum_{S \in \mathbb{S}_{m,n}} |a_S|^2 = 1$.
The action of the linear optics network can be expressed using a matrix $A \in \mathcal{U}_{m,n}$ (where $\mathcal{U}_{m,n}$ is the set of $m \times n$ matrices whose columns are orthonormal).
Then we construct the matrix A_S as a function of the network description $A \in \mathcal{U}_{m,n}$ and a basis state $S = (s_1,\cdots,s_m)$ in the following way: A_S is the $n \times n$ matrix where for $i \in \{1,\cdots,m\}$, we take s_i copies of the i-th row of A.
New we assume that $m \geq n$, and that for the initial state of the system denoted by

$|1_n\rangle$ we have one boson in each of the first n locations (and 0 in the remaining $m-n$ locations):

$$|1_n\rangle = |1\rangle \otimes \cdots \otimes |1\rangle \otimes |0\rangle \otimes \cdots |0\rangle \tag{3.16}$$

Then, it can be proven [AA11] that the target output distribution $\mathcal{D}_A$ over $\mathbb{S}_{m,n}$, indicating the probability to a specific basis state S, after the photons passed through the network described by matrix A, and after measuring the number of photons in each mode, is described by:

$$Pr_{\mathcal{D}_A}[S] = \frac{|\mathsf{Per}(A_S)|^2}{s_1! \cdot s_2! \cdot \cdots \cdot s_m!} \tag{3.17}$$

where $\mathsf{Per}(A_S)$ is the permanent of an the matrix A_S:

$$\mathsf{Per}(A_S) = \sum_{\sigma \in \Sigma_n} \prod_{i=1}^{n} A_{Si,\sigma(i)} \tag{3.18}$$

and Σ_n is the set of all permutations of $\{1, \cdots, n\}$.

As a result Exact Boson sampling is the problem of sampling from the distribution $\mathcal{D}_A$ defined by $Pr_{\mathcal{D}_A}[S]$ when given the input A.

This problem is known to be hard for classical computers and we will present next one of the arguments that support this statement. But first we need to introduce a result known as Stockmeyer approximate counting method:

Theorem 3.2.1. *[From [Sto83]] Let $f : \{0,1\}^n \to \{0,1\}$ be a predicate that can be computed using a deterministic polynomial-time algorithm. Consider the averaged sum:*

$$p = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \tag{3.19}$$

Then, for all $g \geq 1 + \frac{1}{poly(n)}$, there exist a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm that can compute p within a multiplicative factor of g: output $\tilde{p}$, such that $\frac{p}{g} \leq \tilde{p} \leq pg$.

Then, by contradiction suppose there exist a probabilistic polynomial time algorithm M that can solve the Exact Boson Sampling. This is equivalent to M given the input A (the description of the network) can output a sample from the distribution $\mathcal{D}_A$. M being a BPP algorithm can be seen as a deterministic polynomial-time computable function F that when given the input A and an additional string $r \in \{0,1\}^{poly(n)}$ sampled uniformly at random (r can be thought as the internal random coins of M), will output a basis state $S = (s_1, \cdots s_m)$ such that $Pr_{\mathcal{D}_A}[S] = \frac{|\mathsf{Per}(A_S)|^2}{s_1! \cdot s_2! \cdot \cdots \cdot s_m!}$. This can be expressed equivalently as:

$$\Pr_{r \xleftarrow{\$} \{0,1\}^{poly(n)}} [F(A,r) = S] = \frac{|\mathsf{Per}(A_S)|^2}{s_1! \cdot s_2! \cdot \cdots \cdot s_m!} \tag{3.20}$$

Consider now the initial state of the system $|1_n\rangle$. We can compute the probability of obtaining $|1_n\rangle$ in the output modes as:

$$\Pr_{r \xleftarrow{\$} \{0,1\}^{poly(n)}} [F(A,r) = |1_n\rangle] = |\mathsf{Per}(A_{|1_n\rangle})|^2 \tag{3.21}$$

We will define the following predicate f:

$$f(A,r) = \begin{cases} 0, & \text{if } F(A,r) \neq |1_n\rangle \\ 1, & \text{if } F(A,r) = |1_n\rangle \end{cases} \tag{3.22}$$

Now, using this predicate f we can express the probability of outputting $|1_n\rangle$, as the number of random strings r that lead to this output, over the total number of random strings:

$$Pr_{r \xleftarrow{\$} \{0,1\}^{poly(n)}} [F(A,r) = |1_n\rangle] = \frac{1}{2^{poly(n)}} \sum_{r \in \{0,1\}^{poly(n)}} f(A,r) \tag{3.23}$$

Next, we can observe that the predicate f can be computed in polynomial time, since F can be computed in polynomial time and then we just need to check whether the output of F is $|1_n\rangle$.

Then using Stockmeyer Theorem, we can approximate the probability that $F(A,r)$ outputs $|1_n\rangle$ up to multiplicative error using a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm.

In other words, there exist a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm for computing $|\mathsf{Per}(A_{|1_n\rangle})|^2$.

On the other hand in [AA11], it is shown that any matrix $M \in \{-1,0,1\}^{n \times n}$ can be embedded into the input matrix A (using only an extra polynomial overhead) such that $|\mathsf{Per}(A_{|1_n\rangle})|^2$ is proportional to $|\mathsf{Per}(M)|^2$.

As a result, there exist a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm that for any matrix M with elements in $\{-1,0,1\}$, can compute a multiplicative estimate of the permanent of M. But this problem belongs to the class $\#\mathsf{P}$. Additionally, $\mathsf{BPP}^{\mathsf{NP}}$ is contained within the third level of the polynomial hierarchy. Then using Toda's theorem, this implies that the PH would collapse at the third level, which ends the contradiction proof.

3.2.2 GES for Exact Boson Sampling

Having a GES for a general sampling problem means that the Client's input A is a description of a probability distribution $\mathcal{D}_A : \{0,1\}^{poly(|A|)} \to [0,1]$. Then the Client would interact with the Server for a polynomial number of rounds m and finally she would apply the decryption algorithm to obtain: $z := D(\bar{s_m}, k, A)$. Then the outcome z

should, with probability at least $\frac{1}{2} + \frac{1}{poly(|A|)}$ be a sample from the distribution $\mathcal{D}_A$, or in other words:

$$\Pr_{\substack{k \\ \bar{s_m}}} [D(\bar{s_m}, k, A) = z] = Pr[z \leftarrow \mathcal{D}_A] \tag{3.24}$$

From a security point of view, the GES should hide from the Server in an information-theoretic sense everything about A, but its size.

Looking at our case, for the Exact Boson Sampling with m modes and n photons, the input is represented by the $m \times n$ column orthonormal matrix A describing the linear optics network. And the target distribution described by A is $\mathcal{D}_A$, whereas the Client's output S is a sample from $\mathcal{D}_A$ such that: $Pr_{\mathcal{D}_A}[S] = \frac{|\mathsf{Per}(A_S)|^2}{s_1!\cdot s_2!\cdot\cdots\cdot s_m!}$.

The sample S is essentially a particular configuration of the n bosons in the m output locations.

From Lemma 3.1.5, we know that any decision function f that can be delegated in a GES must belong to the class $\mathsf{MA/rpoly}$. The proof of Lemma 3.1.5 can be applied for sampling problems as well.

As a result, if a Client can securely delegate exact sampling from $\mathcal{D}_A$ to a Server using GES, then there exist an $\mathsf{MA/rpoly}$ algorithm for exactly sampling from $\mathcal{D}_A$.

But, very importantly, the result of Lemma 3.1.4 according to which $\mathsf{MA/rpoly} = \mathsf{NP/poly}$ applies only to decision problems, and hence we cannot say this algorithm is equivalent to an $\mathsf{NP/poly}$ sampling algorithm.

In our case, using the fact that $\mathsf{MA/rpoly} \subseteq \mathsf{BPP}^{\mathsf{NP}}\mathsf{/rpoly}$, instead of working with the class $\mathsf{MA/rpoly}$ it will be simpler to consider the sampling algorithm as a $\mathsf{BPP}^{\mathsf{NP}}\mathsf{/rpoly}$ algorithm (due to the connection with Stockmeyer theorem, and the complexity of permanent, as we will see later). In other words, the existence of a GES for sampling problems implies the existence of a probabilistic polynomial-time algorithm with access to an NP oracle and to a randomized polynomial-sized advice that can solve the target sampling problem.

In order to connect this implication concerning the secure delegation of Exact Boson Sampling with the complexity of computing the matrix permanent we must first show some intermediate results about the latter in the next section.

3.2.3 Circuits for the Permanent

The aim of the following intermediate results concerning the analysis of the matrix permanent is to finally show that using an oracle for estimating the squared permanent of a $n \times n$ matrix with values in $\{-1, 0, 1\}$, we can construct a polynomial time algorithm

having random access to $n^{O(n)}$ bits of advice that can exactly compute the permanent. We would use this result together with the assumption that using a GES a Client can delegate the exact sampling from the Boson Sampling distribution and the result of Björklund [Bjö16], to prove our main implication about the existence of GES for sampling problems.

We first introduce the notation: given a matrix A, we will denote with $A^{i,j}$ the matrix obtained from A by eliminating the row i and the column j and with $A^{(i_1,i_2),(j_1,j_2)}$ the matrix obtained from A by eliminating the rows i_1 and i_2 and the columns j_1 and j_2.

Lemma 3.2.2. *Consider am $n \times n$ matrix $X = (x_{i,j}) \in \{-1,0,1\}^{n\times n}$. Then there exists an $(n+2)\times(n+2)$ matrix $Z = (z_{i,j}) \in \{-1,0,1\}^{(n+2)\times(n+2)}$ such that:*

1. $z_{n+2,n+2} = 0$
2. $\mathsf{Per}(Z) = -\mathsf{Per}(X)$
3. $\mathsf{Per}(Z^{n+2,n+2}) = \mathsf{Per}(X^{1,1})$

Proof. We give the following construction of the matrix Z:

$$Z = \begin{pmatrix} x_{n,n} & x_{n,n-1} & \cdots & x_{n,1} & 0 & 0 \\ x_{n-1,n} & x_{n-1,n-1} & \cdots & x_{n-1,1} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{2,n} & x_{2,n-1} & \cdots & x_{2,1} & 0 & 0 \\ x_{1,n} & x_{1,n-1} & \cdots & x_{1,1} & 1 & -1 \\ 0 & 0 & \cdots & 1 & 0 & 1 \\ 0 & 0 & \cdots & -1 & -1 & 0 \end{pmatrix} \tag{3.25}$$

For $\mathsf{Per}(Z^{n+2,n+2})$ using the Laplace development along the last row of $Z^{n+2,n+2}$, and then along the last column of $Z^{(n+2,n+1),(n+2,n)}$ we have:

$$\mathsf{Per}(Z^{n+2,n+2}) = 1 \cdot \mathsf{Per}(Z^{(n+2,n+1),(n+2,n)}) = \mathsf{Per}(X^{1,1}) \tag{3.26}$$

For $\mathsf{Per}(Z)$ using the Laplace development along the last row we have:

$$\mathsf{Per}(Z) = -\mathsf{Per}(Z^{n+2,n+1} - \mathsf{Per}(Z^{n+2,n}) \tag{3.27}$$

But for $\mathsf{Per}(Z^{n+2,n}$ using the development along the lost row we obtain: $\mathsf{Per}(Z^{n+2,n} = \mathsf{Per}(X^{1,1})$ and similarly for $\mathsf{Per}(Z^{n+2,n+1})$ we obtain: $\mathsf{Per}(Z^{n+2,n+1}) = \mathsf{Per}(X) - \mathsf{Per}(X^{1,1})$. Therefore, $\mathsf{Per}(Z) = -\mathsf{Per}(X) + \mathsf{Per}(X^{1,1}) - \mathsf{Per}(X^{1,1}) = -\mathsf{Per}(X)$ which concludes the proof.

□

Lemma 3.2.3. *Consider the matrices $X = (x_{i,j}) \in \{-1,0,1\}^{n\times n}$, $Z = (z_{i,j}) \in \{-1,0,1\}^{m\times m}$, such that $z_{m,m} = 0$ and $m \geq 2$ and the matrix $W = (w_{i,j}) \in \{-1,0,1\}^{(m+n-1)\times(m+n-1)}$ constructed in the following way:*

$$W = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,m} & 0 & \cdots & 0 \\ z_{2,1} & z_{2,2} & \cdots & z_{2,m} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ z_{m-1,1} & z_{m-1,2} & \cdots & z_{m-1,m} & 0 & \cdots & 0 \\ z_{m,1} & z_{m,2} & \cdots & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ 0 & 0 & \cdots & x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{pmatrix} \tag{3.28}$$

Then, we have:

$$\mathsf{Per}(W) = \mathsf{Per}(Z)\cdot\mathsf{Per}(X^{1,1}) + \mathsf{Per}(Z^{m,m})\cdot\mathsf{Per}(X) \tag{3.29}$$

Proof. We will prove this relation by induction over m.
For the base case $m = 2$, we have:

$$W = \begin{pmatrix} z_{1,1} & z_{1,2} & 0 & \cdots & 0 \\ z_{2,1} & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ 0 & x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{pmatrix} \tag{3.30}$$

In this case we have $\mathsf{Per}(Z) = z_{1,2}\cdot z_{2,1}$ and $\mathsf{Per}(Z^{2,2}) = z_{1,1}$.
For W if we use Laplace expansion for the first row, we have:

$$\begin{aligned} \mathsf{Per}(W) &= z_{1,1}\cdot\mathsf{Per}(X) + z_{1,2}\cdot\mathsf{Per}(W^{1,2}) \\ &= z_{1,1}\cdot\mathsf{Per}(X) + z_{1,2}\cdot z_{2,1}\cdot\mathsf{Per}(X^{1,1}) \\ &= \mathsf{Per}(Z^{2,2})\cdot\mathsf{Per}(X) + \mathsf{Per}(Z)\cdot\mathsf{Per}(X^{1,1}) \end{aligned} \tag{3.31}$$

For the inductive step, we assume that the relation holds for m and we want to prove it for dimension $m+1$. We will denote with Z' the matrix for the $m+1$ case (of size $(m+1)\times(m+1)$ and with W' the corresponding matrix (of size $(m+n)\times(m+n)$) using Z'.
Then, by using Laplace expansion along the first row for W' we get:

$$\mathsf{Per}(W') = z_{1,1}\mathsf{Per}(W'^{1,1}) + \cdots + z_{1,m-1}\mathsf{Per}(W'^{1,m-1}) + z_{1,m}\mathsf{Per}(W'^{1,m}) + z_{1,m+1}\mathsf{Per}(W'^{1,m+1}) \tag{3.32}$$

But for any $i \in \{1,...,m\}$ the matrix obtained by eliminating the first row and column i is of the same form and dimension as the matrix W, and therefore we can apply the inductive hypothesis for all the terms in the previous sum but the last one:

$$\begin{aligned}
&\mathsf{Per}(W') = \sum_{i=1}^{m} z_{1,i}\mathsf{Per}(W'^{1,i}) + z_{1,m+1}\mathsf{Per}(W'^{1,m+1}) \\
&= \sum_{i=1}^{m} z_{1,i}\left[\mathsf{Per}(Z'^{1,i}) \cdot \mathsf{Per}(X^{1,1}) + \mathsf{Per}(Z'^{(1,m+1),(i,m+1)}) \cdot \mathsf{Per}(X)\right] + z_{1,m+1}\mathsf{Per}(W'^{1,m+1}) \\
&= \mathsf{Per}(X^{1,1})\left[\sum_{i=1}^{m} z_{1,i}\mathsf{Per}(Z'^{1,i})\right] + \mathsf{Per}(X)\left[\sum_{i=1}^{m} z_{1,i}\mathsf{Per}(Z'^{(1,m+1),(i,m+1)})\right] + z_{1,m+1}\mathsf{Per}(W'^{1,m+1})
\end{aligned} \tag{3.33}$$

But we know that $\mathsf{Per}(Z'^{m+1,m+1}) = \sum_{i=1}^{m} z_{1,i} \cdot \mathsf{Per}(Z'^{(1,m+1),(i,m+1)})$ (by using Laplace expansion after the first row). Hence, we can rewrite $\mathsf{Per}(W')$ as:

$$\mathsf{Per}(W') = \mathsf{Per}(X^{1,1})\left[\sum_{i=1}^{m} z_{1,i}\mathsf{Per}(Z'^{1,i})\right] + \mathsf{Per}(X) \cdot \mathsf{Per}(Z'^{m+1,m+1}) + z_{1,m+1}\mathsf{Per}(W'^{1,m+1}) \tag{3.34}$$

Now, we take separately the terms from this expression.

$$W'^{1,m+1} = \begin{pmatrix}
z_{2,1} & z_{2,2} & \cdots & z_{2,m} & 0 & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
z_{m,1} & z_{m,2} & \cdots & z_{m,m} & 0 & \cdots & 0 \\
z_{m+1,1} & z_{m+1,2} & \cdots & z_{m+1,m} & x_{1,2} & \cdots & x_{1,n} \\
0 & 0 & \cdots & 0 & x_{2,2} & \cdots & x_{2,n} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & 0 & x_{n,2} & \cdots & x_{n,n}
\end{pmatrix} \tag{3.35}$$

We observe that $W'^{1,m+1}$ is of the same form as W with the following instantiations for $\bar{X}$ and $\bar{Z}$:

$$\begin{aligned}
\bar{Z} &= \begin{pmatrix}
z_{2,1} & z_{2,2} & \cdots & z_{2,m} \\
\cdots & \cdots & \cdots & \cdots \\
z_{m,1} & z_{m,2} & \cdots & z_{m,m} \\
z_{m+1,1} & z_{m+1,2} & \cdots & 0
\end{pmatrix} \\
\bar{X} &= \begin{pmatrix}
z_{m+1,m} & x_{1,2} & \cdots & x_{1,n} \\
0 & x_{2,2} & \cdots & x_{2,n} \\
\cdots & \cdots & \cdots & \cdots \\
0 & x_{n,2} & \cdots & x_{n,n}
\end{pmatrix}
\end{aligned} \tag{3.36}$$

Using again the induction hypothesis, we have that:

$$\mathsf{Per}(W'^{1,m+1}) = \mathsf{Per}(\bar{Z}) \cdot \mathsf{Per}(\bar{X}^{1,1}) + \mathsf{Per}(\bar{Z}^{m,m}) \cdot \mathsf{Per}(\bar{X}) \tag{3.37}$$

But by decomposing $\mathsf{Per}(X)$ after the first column we have that:

$$\mathsf{Per}(\bar{X}) = z_{m+1,m} \cdot \mathsf{Per}(X^{1,1})$$

Additionally, we observe that $\bar{X}^{1,1} = X^{1,1}$, and therefore we have that:

$$\begin{aligned} \mathsf{Per}(W'^{1,m+1}) &= \mathsf{Per}(\bar{Z}) \cdot \mathsf{Per}(X^{1,1}) + z_{m+1,m} \cdot \mathsf{Per}(\bar{Z}^{m,m}) \cdot \mathsf{Per}(X^{1,1}) \\ &= \mathsf{Per}(X^{1,1})[\mathsf{Per}(\bar{Z}) + z_{m+1,m} \cdot \mathsf{Per}(\bar{Z}^{m,m})] \end{aligned} \tag{3.38}$$

And now, if we analyze the $\bar{Z}$ component, we have:
Firstly, we observe that $\bar{Z}^{m,m} = Z^{1,m}$. Secondly, let us analyze the two matrices $Z'^{1,m+1}$ and $\bar{Z}$:

$$Z'^{1,m+1} = \begin{pmatrix} z_{2,1} & z_{2,2} & \cdots & z_{2,m} \\ \cdots & \cdots & \cdots & \cdots \\ z_{m,1} & z_{m,2} & \cdots & z_{m,m} \\ z_{m+1,1} & z_{m+1,2} & \cdots & z_{m+1,m} \end{pmatrix} \tag{3.39}$$

If we develop the permanents of $Z'^{1,m+1}$ and $\bar{Z}$ after the last line we obtain:

$$\begin{aligned} \mathsf{Per}(Z'^{1,m+1}) &= \left[\sum_{i=1}^{m-1} z_{m+1,i} \cdot \mathsf{Per}(Z^{1,i})\right] + z_{m+1,m} \cdot \mathsf{Per}(Z^{1,m}) \\ \mathsf{Per}(\bar{Z}) &= \sum_{i=1}^{m-1} z_{m+1,i} \cdot \mathsf{Per}(Z^{1,i}) \end{aligned} \tag{3.40}$$

Thus by combining the 3 previous equations we get:

$$\mathsf{Per}(Z'^{1,m+1}) = \mathsf{Per}(\bar{Z}) + z_{m+1,m} \cdot \mathsf{Per}(Z^{1,m}) = \mathsf{Per}(\bar{Z}) + z_{m+1,m} \cdot \mathsf{Per}(\bar{Z}^{m,m}) \tag{3.41}$$

Then by replacing this in Equation 3.38, we obtain:

$$\mathsf{Per}(W'^{1,m+1}) = \mathsf{Per}(X^{1,1}) \cdot \mathsf{Per}(Z'^{1,m+1}) \tag{3.42}$$

Now, finally we can go back to $Per(W')$ and by replacing in Equation 3.34 the previous expression of $\mathsf{Per}(W'^{1,m+1})$, we derive:

$$\begin{aligned} \mathsf{Per}(W') &= \mathsf{Per}(X^{1,1}) \left[\sum_{i=1}^{m} z_{1,i} \mathsf{Per}(Z'^{1,i})\right] + \mathsf{Per}(X) \cdot \mathsf{Per}(Z'^{m+1,m+1}) + \\ &+ z_{1,m+1} \mathsf{Per}(X^{1,1}) \cdot \mathsf{Per}(Z'^{1,m+1}) \\ &= \mathsf{Per}(X^{1,1}) \cdot \left[\sum_{i=1}^{m+1} z_{1,i} \mathsf{Per}(Z'^{1,i})\right] + \mathsf{Per}(X) \cdot \mathsf{Per}(Z'^{m+1,m+1}) \\ &= \mathsf{Per}(X^{1,1}) \cdot \mathsf{Per}(Z') + \mathsf{Per}(X) \cdot \mathsf{Per}(Z'^{m+1,m+1}) \end{aligned} \tag{3.43}$$

which concludes the inductive proof. □

Using 2 general results about permanents we can now show that by having an oracle that gives us a multiplicative approximation of the squared permanent of an $(n \times n)$ matrix, we can construct a polynomial-time algorithm with $n^{O(n)}$ bits of advice that can exactly compute the permanent of the matrix.
The proof of the following implication is inspired from the result of Aaronson and Arkhipov (Thm. 4.3 of [AA11]). However, in their case the oracle was outputting a multiplicative approximation of the squared permanent of a matrix with arbitrary real elements, while in our case the matrices are restricted to elements from $\{-1,0,1\}$.

Theorem 3.2.4. *Consider $\mathbb{O}$ an oracle such that when queried with input a matrix $X \in \{-1,0,1\}^{n\times n}$, it outputs a value $\mathbb{O}(X)$ satisfying:*

$$\frac{\mathsf{Per}(X)^2}{g} \leq \mathbb{O}(X) \leq g\mathsf{Per}(X)^2 \ \text{, where } g \in [1, poly(n)] \tag{3.44}$$

Then for any matrix $X \in \{-1,0,1\}^{n\times n}$ we can compute the permanent of X, using a polynomial time algorithm which has access to $n^{O(n)}$ bits of advice and that can makes $poly(n)$ queries to $\mathbb{O}$.

Proof. We will proceed with a proof by induction over n.
For the base case $n = 1$, we have $\mathsf{Per}(X) = X$ and thus the algorithm can directly output the input X.
For the inductive step we assume for any matrix $X \in \{-1,0,1\}^{n\times n}$ there exist an algorithm $\mathcal{A}$ using $n^{O(n)}$ bits of advice and making $poly(n)$ queries to $\mathbb{O}$, that can compute $\mathsf{Per}(X)$. And we want show that for any $X' \in \{-1,0,1\}^{n\times n}$, we can construct an algorithm $\mathcal{A}'$ that using $(n+1)^{O(n+1)}$ bits of advice and $poly(n+1)$ queries to $\mathbb{O}$ that can output $\mathsf{Per}(X')$.
Now let us examine how we can computer $\mathsf{Per}(X')$.
Firstly, if we query $\mathbb{O}$ and we obtain $\mathbb{O}(X') = 0$, then this happens if and only if $\mathsf{Per}(X') = 0$ ($\frac{\mathsf{Per}(X')^2}{g} \leq 0$, with $g \geq 1$, hence $\mathsf{Per}(X') = 0$) and we are done.
Secondly, we will query $\mathbb{O}$ for all matrices $X'^{i,j}$ to obtain all minors of order n. Now, as $\mathsf{Per}(X) \neq 0$, then there must exist a minor different from 0: exists $i, j \in \{1, \cdots n+1\}$ such that $\mathsf{Per}(X^{i,j}) \neq 0$ (and therefore $\mathbb{O}(X^{i,j})$).
Then, as the permanent is invariant under the permutation of rows and columns, we can consider $(i, j) = (1,1)$. Therefore, we have $\mathsf{Per}(X') \neq 0$ and $\mathsf{Per}(X'^{1,1}) \neq 0$.
Now, if we take a matrix $Z \in \{-1,0,1\}^{m\times m}$, with $m \geq 2$ and $z_{m,m} = 0$, then from Lemma 3.2.3, we know there exists a matrix $W \in \{-1,0,1\}^{(m+n)\times(m+n)}$ such that:

$$\mathsf{Per}(W) = \mathsf{Per}(Z) \cdot \mathsf{Per}(X'^{1,1}) + \mathsf{Per}(Z^{m,m}) \cdot \mathsf{Per}(X') \tag{3.45}$$

If $\mathsf{Per}(W) = 0$ and $\mathsf{Per}(Z^{m,m}) \neq 0$, then we have:

$$\mathsf{Per}(X') = -\frac{\mathsf{Per}(Z) \cdot \mathsf{Per}(X'^{1,1})}{\mathsf{Per}(Z^{m,m})} \tag{3.46}$$

We first show there exist matrices W and Z such that Equation 3.45 holds and such that $\mathsf{Per}(W) = 0$ and $\mathsf{Per}(Z^{m,m}) \neq 0$.
From Lemma 3.2.2 we know that there exists a matrix $Z \in \{-1,0,1\}^{(n+3)\times(n+3)}$ such that $z_{n+3,n+3} = 0$ and:

$$\begin{aligned} \mathsf{Per}(Z) &= -\mathsf{Per}(X') \\ \mathsf{Per}(Z^{n+3,n+3}) &= \mathsf{Per}(X'^{1,1}) \end{aligned} \tag{3.47}$$

where as $\mathsf{Per}(X')$ and $\mathsf{Per}(X'^{1,1})$ are non-zero, then $\mathsf{Per}(Z) \neq 0$ and $\mathsf{Per}(Z^{n+3,n+3}) \neq 0$.
If we set $m = n+3$ and by using this Z in the construction this Z in the construction of W, we have:

$$\mathsf{Per}(W) = -\mathsf{Per}(X') \cdot \mathsf{Per}(X'^{1,1}) + \mathsf{Per}(X'^{1,1}) \cdot \mathsf{Per}(X') = 0 \tag{3.48}$$

Then, we know that for any $X' \in \{-1,0,1\}^{(n+1)\times(n+1)}$ there exist matrices $Z \in \{-1,0,1\}^{(n+3)\times(n+3)}$, with $\mathsf{Per}(Z^{n+3,n+3}) \neq 0$ such that:

$$\mathsf{Per}(X') = -\mathsf{Per}(X^{1,1}) \cdot \frac{\mathsf{Per}(Z)}{\mathsf{Per}(Z^{n+3,n+3})} \tag{3.49}$$

The algorithm $\mathcal{A}'$ will search for a matrix Z, it will construct the corresponding matrix W (as in Lemma 3.2.3) and use the oracle $\mathbb{O}$ to test if $\mathsf{Per}(W) = 0$.
Then if $\mathsf{Per}(W) = 0$, then we can compute $\mathsf{Per}(X')$ using Equation 3.49.
To compute $\mathsf{Per}(X^{1,1})$ we will use $\mathcal{A}$ from the induction hypothesis, but the main question is how to search for the matrix Z and how to compute the factor $\frac{\mathsf{Per}(Z)}{\mathsf{Per}(Z^{n+3,n+3})}$.
Here is where the advice steps in.
Specifically, we will define the advice as the set of tuples:
$\left(Z^{(i)}, \mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right), \frac{\mathsf{Per}\left(Z^{(i)}\right)}{\mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right)}\right)$, where $Z^{(i)}$ are matrices, $Z^{(i)} \in \{-1,0,1\}^{n+3,n+3}$, such that $Z^{(i)}_{n+3,n+3} = 0$ and $\mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right) \neq 0$. But now, we need to see how to choose these matrices $Z^{(i)}$ and how many would be needed to be part of advice.
Firstly, we observe that if $Z^{(i)} \in \{-1,0,1\}^{(n+3)\times(n+3)}$, then $\mathsf{Per}\left(Z^{(i)}\right) \in \mathbb{Z}$ and $-(n+3)! \leq \mathsf{Per}\left(Z^{(i)}\right) \leq (n+3)!$. Using that $2(n+3)! < n^{O(n)}$, we have that there are at most $n^{O(n)}$ possible values for the permanents of $Z^{(i)}$ matrices.
Similarly, as $\mathsf{Per}\left(Z^{(i)}\right)$ can take $2(n+3)!$ values and $\mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right)$ can take $2(n+$

2)!, then $f_i := \frac{\mathsf{Per}\left(Z^{(i)}\right)}{\mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right)}$ can take at most $n^{O(n)}$ different values.

As a result, the matrices $Z^{(i)}$ that constitute the advice will be chosen such that all possible values of $f_i = \frac{\mathsf{Per}\left(Z^{(i)}\right)}{\mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right)}$ are covered. Hence, the number of tuples:

$(Z^{(i)}, \mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right), f_i)$ is at most $n^{O(n)}$. Each tuple requires $2(n+3)^2 + [\log(2(n+3)!)] + [\log(2(n+3)!(n+2)!)] = O(n^2)$. As a result the size of the advice will be $n^{O(n)} \cdot O(n^2) = n^{O(n)}$.

Furthermore, we place the tuples in the advice in ascending order with respect to f_i.

Then for a given input X', our algorithm $\mathcal{A}'$, will search for the advice in order to find a matrix $Z^{(i)}$ such that $\mathsf{Per}(W^{(i)}) = 0$ (constructed as in Lemma 3.2.3), which $\mathcal{A}'$ will test by running $\mathbb{O}(W^{(i)}) = 0$.

Then, when such a matrix $Z^{(i)}$ is found we will use the corresponding f_i to compute the permanent of X' as:

$$\mathsf{Per}(X') = -\mathsf{Per}(X^{1,1}) \cdot f_i \tag{3.50}$$

Finally, to computer $\mathsf{Per}(X^{1,1})$, as $X^{1,1}$ is an $n \times n$ matrix, we can run the algorithm $\mathcal{A}$ from the inductive hypothesis.

Therefore, the last thing we need to take care of, is how to search in the advice for a matrix $Z^{(i)}$ such that $\mathsf{Per}(W^{(i)}) = 0$.

Suppose i ranges from 1 to $l = n^{O(n)}$ and let us denote $\alpha_i = \mathsf{Per}\left(Z^{(i)^{n+3,n+3}}\right)$. Then for any $i \in \{1 \cdots, l\}$, we have:

$$\mathsf{Per}(W^{(i)}) = \mathsf{Per}(Z^{(i)}) \cdot \mathsf{Per}(X'^{1,1}) + \alpha_i \cdot \mathsf{Per}(X') \tag{3.51}$$

Equivalent to:

$$\mathsf{Per}(W^{(i)}) = \alpha_i(f_i \cdot \mathsf{Per}(X'^{1,1}) + \mathsf{Per}(X')) \tag{3.52}$$

Then, by running $\mathbb{O}(W^{(i)})$ we obtain a multiplicative approximation of $\mathsf{Per}(W^{(i)})^2$:

$$\begin{aligned} &\frac{\mathsf{Per}(W^{(i)})^2}{g_i} \leq \mathbb{O}(W^{(i)}) \leq g_i \mathsf{Per}(W^{(i)})^2 \ , \ g_i \in [1, poly(n)] \\ &\frac{|\mathsf{Per}(W^{(i)})|}{\sqrt{g_i}} \leq \sqrt{\mathbb{O}(W^{(i)})} \leq \sqrt{g_i}|\mathsf{Per}(W^{(i)})| \end{aligned} \tag{3.53}$$

And hence as α_i is part of the advice, we can obtain a multiplicative approximation of $|f_i \cdot \mathsf{Per}(X'^{1,1}) + \mathsf{Per}(X')|$, by computing $\frac{\sqrt{\mathbb{O}(W^{(i)})}}{\alpha_i}$:

$$\frac{1}{\sqrt{g_i}}|f_i \cdot \mathsf{Per}(X'^{1,1}) + \mathsf{Per}(X')| \leq \frac{\sqrt{\mathbb{O}(W^{(i)})}}{\alpha_i} \leq \sqrt{g_i}|f_i \cdot \mathsf{Per}(X'^{1,1}) + \mathsf{Per}(X')| \tag{3.54}$$

If we consider the function $h(i) := \mathsf{Per}(X') + f_i \mathsf{Per}(X'^{1,1})$ as a function of index $i \in \{1, \cdots, l\}$, then as f_i is an increasing sequence, this means that h is a strictly increasing function if $\mathsf{Per}(X'^{1,1}) > 0$ and strictly decreasing otherwise. Hence h is a strictly monotone and thus injective function. As we showed before, in Equation 3.49 there exists $\bar{i}$ such that $h(\bar{i}) = 0$. As h is injective it means that this solution is unique.
Therefore our goal is to find the unique $\bar{i} \in \{1, \cdots, l\}$ such that $h(\bar{i}) = 0$. We have a multiplicative approximation for $|h(i)|$, which we denote as $t(i) := \frac{\sqrt{\mathbb{O}(W^{(i)})}}{\alpha_i}$. This function will be strictly decreasing between 1 and $\bar{i}$ and strictly increasing between $\bar{i}$ and l. We search $\bar{i}$ using binary search as follows: we choose 2 middle points v and w that divide the interval $[1, l]$. If $t(v) = 0$ or $t(w) = 0$ then we are done. Otherwise, if $t(v) < t(w)$, then we search on the interval $[2, v]$, otherwise on the interval $[w+1, l]$. And we repeat this recursively, until the minimum is found.
Given that the number of tuples is $n^{O(n)}$, the algorithm will query $\mathbb{O}$ at most $O(n \log(n))$ times. Additionally, the construction of each matrix $W^{(i)}$ requires $O(n^2)$ operations and since this computation is performed $O(n \log(n))$ times, the complexity of this step is $O(n^3 \log(n))$. Then $\mathcal{A}'$ calls $\mathcal{A}$ once to computer $\mathsf{Per}(X)$. $\mathcal{A}$ in turn requires $n^{O(n)}$ bits of advice and performs $\mathsf{poly}(n)$ queries to $\mathbb{O}$. Thus, $\mathcal{A}'$ uses $n^{O(n)}$ bits of advice and queries $\mathbb{O}$ for $\mathsf{poly}(n)$ times, which concludes the proof.

□

Theorem 3.2.5. *If there exists a* $\mathsf{BPP/rpoly}$ *algorithm that can sample exactly from the Boson Sampling distribution, then for any matrix* $X \in \{-1, 0, 1\}^{n \times n}$*, there exist circuits of size* $2^{n - \Omega\left(\frac{n}{\log(n)}\right)}$*, making polynomially many queries to an* NP *oracle for computing* $\mathsf{Per}(X)$.

Proof. The starting point of the proof is a result of Björklund according to which for any $k \leq n$, the permanent of an $n \times n$ matrix X can be expressed as a linear combination of $\mathsf{poly}(n) \cdot 2^{n-k}$ many permanents of $k \times k$ matrices. It should be noted, that while these $k \times k$ matrices are not necessarily minors of the original matrix X, they can still be computed efficiently given X.
Our task is to compute all of these $\mathsf{poly}(n) 2^{n-k}$ permanents and then compute their linear combination to obtain the permanent of X.
To do so, we will use the result of Theorem 3.2.4 together with the assumption of this theorem that there exists a $\mathsf{BPP/rpoly}$ for Exact Boson Sampling, to show that the permanent of any $k \times k$ matrix can be computed in polynomial time using random access to $k^{O(k)}$ bits of advice and polynomially-sized queries to an NP oracle.

Crucially, the $k^{O(k)}$-sized advice will be the same for all $k \times k$ matrices. This mean that to compute all the $k \times k$ permanents, and thus to compute $\mathsf{Per}(X)$, we can do it in $\mathsf{poly}(n)\,2^{n-k}$ time with access to $k^{O(k)}$ bits of advice.

The explicit value of k as a function of n will be determined later.

Consider a $k \times k$ matrix M and a parameter $\varepsilon \in (0,1)$. We will embed εM, a scaled version of M, as a submatrix of a Boson Sampling input $A_{\varepsilon M}$, as shown in [AA11].

In other words, $A_{\varepsilon M} \in \mathbb{C}^{m \times k}$, with $m = \mathsf{poly}(k)$. Then, the probability of obtaining the state $|1_k\rangle$ in the output mode is:

$$p = \mathsf{Per}(\varepsilon M)^2 = \varepsilon^{2k} \cdot \mathsf{Per}(M)^2 \tag{3.55}$$

Since $\mathsf{Per}(M)^2 \leq (k!)^2$, from $p \leq 1$, we require $\varepsilon \leq \frac{1}{\sqrt[k]{k!}}$, hence we can choose $\varepsilon = \frac{1}{k}$.

If a $\mathsf{BPP}/\mathsf{rpoly}$ algorithm can solve Exact Boson Sampling with the input network description $A_{\varepsilon M}$, then there exists a BPP algorithm $\mathcal{A}$ and a probability distribution $\mathcal{D}_k = \{q_y\}_{y \in \{0,1\}^{\mathsf{poly}(k)}}$ which only depends on k, such that:

$$\sum_{y \in \{0,1\}^{\mathsf{poly}(k)}} q_y \cdot \Pr\{\mathcal{A}(A_{\varepsilon M}, y) = |1_k\rangle\} = p = \frac{\mathsf{Per}(M)^2}{k^{2k}} \tag{3.56}$$

where y represents the rpoly advice sampled according to $\mathcal{D}_k$.

It is clear that if we can estimate the probability p with multiplicative error in polynomial time (potentially using an NP oracle and $k^{O(k)}$ bits of advice then we can simulate the behaviour of the oracle $\mathbb{O}$ from Theorem 3.2.4 and then compute $\mathsf{Per}(M)$ using a polynomial time algorithm with $k^{O(k)}$ bits of advice and making $\mathsf{poly}(k)$ queries to $\mathbb{O}$.

Now, as M has elements in $\{-1,0,1\}$, then $\mathsf{Per}M \in \mathbb{Z}$ and $-k! \leq \mathsf{Per}(M) \leq k!$. If $\mathsf{Per}(M) \neq 0$, then we have: $1 \leq \mathsf{Per}(M)^2 \leq (k!)^2$. Hence:

$$\frac{1}{k^{2k}} \leq p \leq \frac{(k!)^2}{k^{2k}} \tag{3.57}$$

Moreover, as $p = \frac{\mathsf{Per}(M)^2}{k^{2k}}$, p can take at most $(k!)^2 < k^{O(k)}$ different values.

Then, the advice string S will consist of $k^{O(k)}$ samples from $\mathcal{D}_k$ together with their probabilities: $S = \{(y_i, q_{y_i})\}_{1 \leq i \leq k^{O(k)}}$. Using this advice we can define:

$$p_{est} = \sum_{y \in S} q_y \cdot \Pr\{\mathcal{A}(A_{\varepsilon M}, y) = |1_k\rangle\} \tag{3.58}$$

as a multiplicative estimate of p. Since $\mathcal{A}$ is a BPP algorithm, then as before, we can view it as a deterministic polynomial-time computable function f_A that receives as input $A_{\varepsilon M}, y$ and an additional random string $r \in \{0,1\}^{l(k)}$, with l polynomial. The

function f_A will output 1 when $\mathcal{A}$ outputs $|1_k\rangle$ and will output 0 otherwise. Hence, we have:

$$Pr_{r \xleftarrow{\$} \{0,1\}^{l(k)}}[\mathcal{A}(A_{\varepsilon M}, y) = |1_k\rangle] = \frac{1}{2^{l(k)}} \sum_{r \in \{0,1\}^{l(k)}} f_A(A_{\varepsilon M}, y, r) \tag{3.59}$$

And therefore the estimate for p is equal to:

$$p_{est} = \frac{1}{2^{l(k)}} \sum_{r \in \{0,1\}^{l(k)}} \sum_{y \in S} q_y \cdot f_A(A_{\varepsilon M}, y, r) \tag{3.60}$$

However, it can be seen that computing p_{est} requires summing $k^{O(k)} 2^{l(k)}$ terms, each of them being computed efficiently in polynomial time using the deterministic function f_A and the advice.

Therefore, we will use Stockmeyer Theorem (Thm 3.2.1) in order to obtain a multiplicative estimate of p_{est}. Consequently, there exists a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm that can compute p_{est} up to a multiplicative error. This will then also yield a multiplicative approximation of p.

More specifically, we showed that to compute the multiplicative estimate of a $k \times k$ matrix M, we can use a BPP algorithm having access to $k^{O(k)}$ bits of advice and to an NP oracle (for Stockmeyer counting). But then, this algorithm can be seen as an implementation of the oracle $\mathbb{O}$.

Therefore, from Theorem 3.2.4, we have an algorithm for computing the permanent of M *exactly*, running in polynomial time using $k^{O(k)}$ bits of advice and access to an NP oracle.

But very importantly, as the advice is the same for all $k \times k$ matrices, we can then repeat the same procedure for all $\mathsf{poly}(n) \cdot 2^{n-k}$ permanents.

This yields that in order to compute $\mathsf{Per}(X)$ we require $\mathsf{poly}(n) \cdot 2^{n-k} \cdot \mathsf{poly}(k)$ operations and access to $k^{O(k)}$ bits of advice and to an NP oracle.

Finally, we need to convert this algorithm into a circuit.

We can now choose k conveniently that would minimize both the number of operations $\mathsf{poly}(n) \cdot 2^{n-k} \cdot \mathsf{poly}(k)$ and the size of the advice $k^{O(k)}$.

Then if we choose $k = c \cdot \frac{n}{\log n}$, for some constant $c > 0$, we have:

The number of operations becomes: $\mathsf{poly}(n) \cdot \mathsf{poly}\left(\frac{n}{\log n}\right) \cdot 2^{n - c \cdot \frac{n}{\log n}} = 2^{n - \Omega\left(\frac{n}{\log n}\right)}$.

And the size of advice: $\left(\frac{cn}{\log n}\right)^{\frac{c'n}{\log n}} = 2^{n - \Omega\left(\frac{n}{\log n}\right)}$. Therefore, we have circuits of size $2^{n - \Omega\left(\frac{n}{\log n}\right)}$. And to implement the random access to the $2^{n - \Omega\left(\frac{n}{\log n}\right)}$ bits of advice we consider that the gates have unbounded fan-in. Hence, the advice bits are hardcoded into the circuit and fed whenever the algorithm $\mathcal{A}$ uses them. Since only polynomially

many bits of advice are used at any given step of the algorithm, this will increase the size of the circuit by a factor polynomial in n, which still keeps the size of the circuit as $2^{n-\Omega\left(\frac{n}{\log n}\right)}$, concluding the proof.

□

We are now ready to show our main result about the secure delegation of the Boson Sampling problem.

Theorem 3.2.6. *If Exact Boson Sampling admits a GES then for any matrix $X \in \{-1,0,1\}^{n\times n}$ there exist circuits of size $2^{n-\Omega\left(\frac{n}{\log n}\right)}$ making poly-sized queries to an $\mathsf{NP}^{\mathsf{NP}}$ oracle for computing $\mathsf{Per}(X)$.*

Proof. If Boson Sampling admits a GES then as shown in subsection 3.2.2 this implies that there exist a BPP algorithm with access to an NP oracle and to a randomized poly-sized advice that can sample from the Boson Sampling distribution $\mathcal{D}_A$.
The result of Theorem 3.2.5 relativises with respect to an NP oracle. More specifically, this implies that if a $\mathsf{BPP}^{\mathsf{NP}}/\mathsf{rpoly}$ algorithm can sample exactly from the Boson Sampling distribution then for any matrix $X \in \{-1,0,1\}^{n\times n}$ there exist circuits of size $2^{n-\Omega\left(\frac{n}{\log n}\right)}$ making poly-sized queries to an $\mathsf{NP}^{\mathsf{NP}}$ oracle that can compute the permanent of X. □

Chapter 4

QFactory against Honest-but-Curious Server

Secure delegated quantum computation between a classical client and a quantum server with information theoretic security is implausible given our results in the previous chapter. As a result, we turn our attention to achieving the same task under more restricted levels of security. Specifically, the question we address in this chapter is obtaining classical delegation of quantum computations with post-quantum computational security against a malicious Server.

To achieve this task, the key is a primitive that allows us to replace the need for (a particular) quantum communication channel with a computationally (but post-quantum) secure generation of secret and random qubits using exclusively classical resources. This can be used by classical clients to achieve blind quantum computing but also, because of the modularity of the functionality, can be used in a number of other applications too (such as multi-party quantum computation).

We call this primitive *classical client remote state preparation (*$\mathsf{CC-RSP}$*)*, where a classical client can instruct the preparation of a sequence of random qubits at some distant party. Their classical description is (computationally) unknown to any other party (including the distant party preparing them) but known to the client. We emphasize the unique feature that no quantum communication is required to implement $\mathsf{CC-RSP}$. This enables classical clients to perform a class of quantum communication protocols with only a public classical channel between the classical clients and a quantum server. One of the main example is in fact our goal, purely classical-client (computational) secure delegation of quantum computations.

In this chapter we will give a concrete protocol called $\mathsf{HBC-QFactory}$ imple-

menting $\mathsf{CC}-\mathsf{RSP}$, using the Learning-With-Errors problem to construct a trapdoor one-way function with certain desired properties (quantum-safe, two-regular, collision-resistant). We then prove the security in the game-based framework, in the semi-honest setting.

The $\mathsf{CC}-\mathsf{RSP}$ primitive, viewed as a resource, by replacing the need for quantum channel between parties in certain quantum communication protocols with trade-off that the protocols become computationally secure (against *quantum* adversaries), has a wide range of applications. Here we will present a general overview of the applications of $\mathsf{CC}-\mathsf{RSP}$.

The first category of applications concerns a large class of delegated quantum computation protocols, including blind quantum computation and *verifiable* blind quantum computation. These protocols are of great importance, enabling information-theoretically secure (and verifiable) access to a quantum cloud. However, the requirement for quantum communication limits their domain of applicability. This limitation is removed by replacing the off-line preparation stage with our QFactory protocol. Concretely, we can use QFactory to implement the blind quantum computation protocol of [BFK09], as well as the *verifiable* blind quantum computation protocols (e.g. those in [FK17, Bro15b, FKD17]), in order to achieve classical-client secure and verifiable access to a quantum cloud.

The second category of applications refers a more general family of protocols for which their quantum communication consists of random single qubits similar to those provided by our protocol $\mathsf{HBC}-\mathsf{QFactory}$, such as: quantum-key-distribution [BB14], quantum money [BOV$^+$18], quantum coin-flipping [PCDK11], quantum signatures [WDKA15], etc.

In this chapter **our contributions** can be summarized as follows:

1. We define the primitive *Classical Client Remote State Preparation* ($\mathsf{CC}-\mathsf{RSP}_\theta$[1]) in section 4.2. $\mathsf{CC}-\mathsf{RSP}_\theta$ can replace the need for quantum channel between parties in certain quantum communication protocols with trade-off that the protocols become computationally secure (against *quantum* adversaries).

2. We give a basic protocol ($\mathsf{HBC}-\mathsf{QFactory}$) that achieves this functionality from a correctness point of view, given a trapdoor one-way function that is quantum-

[1] the parameter θ refers to the set of quantum states produced by this primitive, which are the quantum states $\{|+_\theta\rangle\}_{\theta\in\{0,\cdots,7\pi/4\}}$

safe, two-regular and collision resistant in section 4.3 and prove its correctness.

3. We prove the security of the $\mathsf{HBC-QFactory}$ against Honest-But-Curious server (server follows the protocol specifications, but can try to infer any information about the secret from the classical transcripts) or against any malicious third party using a game based security definition. To show the security we prove that the classical description of the generated qubits is a hard-core function (following a reduction similar that of the Goldreich-Levin Theorem) in section 4.4.

4. While the above-mentioned results do not depend on the specific function used, the existence of such specific functions (with all desired properties) makes the $\mathsf{CC-RSP}_\theta$ a practical primitive that can be employed as described in this paper. In section 4.5, we first give methods for obtaining two-regular trapdoor one-way functions with extra properties (collision resistant or second preimage resistant) assuming the existence of simpler trapdoor one-way functions (permutation trapdoor or homomorphic trapdoor functions). We use reductions to prove that the resulting functions maintain all the properties required. Furthermore, we give in subsection 4.5.3 an explicit family of functions that respect all the required properties based on the security of the Learning-With-Errors problem as well as a possible instantiation of the parameters. This function is also quantum-safe, and thus directly applicable for our setting. Note, that other functions may also be used, such as the one in [BCM$^+$18] or functions based on the Niederreither cryptosystem and the construction in [FGK$^+$10].

5. Finally, we implement $\mathsf{HBC-QFactory}$ on the quantum computer IBM Quantum Experience using a toy function (given the current limited number of available qubits we consider a 2-regular function acting on a small number of bits, consequently, it cannot be post-quantum secure). Hence, we provide in addition to the theoretical results, an experimental evidence of the correctness and output distribution of the $\mathsf{HBC-QFactory}$ Protocol on a real quantum device.

4.1 Overview of the Protocol and Proof

The general idea is that a classical client gives instructions to a quantum server to perform certain actions (quantum computation). Those actions lead to the server having as output a single qubit, which is randomly chosen from within a set of possible states

of the form $|+_\theta\rangle := 1/\sqrt{2}(|0\rangle + e^{i\theta}|1\rangle)$, where $\theta \in \{0, \frac{\pi}{4}, \cdots, \frac{7\pi}{4}\}$. The randomness of the output qubit is due to the (fundamental) randomness of quantum measurements that are part of the instructions that the client gives. Moreover, the server cannot guess the value of θ any better than if he had just received that state directly from the client (up to negligible probability). This is possible because the instructed quantum computation is generically a computation that is hard to (i) classically simulate and (ii) to reproduce quantumly because it is unlikely (exponentially in the number of measurements) that by running the same instructions the server obtains the exact same measurement outcomes twice. On the other hand, we wish the client to *know* the classical description and thus the value of θ. To achieve this task, the instructions/quantum computation the client uses are based on a family of trapdoor one-way functions with certain extra properties[2]. Such functions are hard to invert (e.g. for the server) unless someone (the client in our case) has some extra "trapdoor" information t_k. This extra information makes the quantum computation easy to classically reproduce for the client, which can recover the value θ, while it is still hard to classically reproduce for the server. Sending random qubits of the above type, is exactly what is required from the client in most of the protocols and applications given earlier, while with simple modifications our protocol could achieve other similar sets of states.

Our $\mathsf{HBC-QFactory}$ protocol can heuristically be described in the next steps:

Preparation. The client randomly selects a function f_k, from a family of trapdoor one-way, quantum-safe, two-regular and collision resistant functions. The choice of f_k is public (server knows), but the trapdoor information t_k needed to invert the function is known only to the client.

Stage 1: Preimages Superposition. The client instructs the server (i) to apply Hadamard(s) on the control register, (ii) to apply U_{f_k} on the target register i.e. to obtain $\sum_x |x\rangle \otimes |f_k(x)\rangle$ and (iii) to measure the target register in the computational basis, in order to obtain a value y. This collapses his state to the state $(|x\rangle + |x'\rangle) \otimes |y\rangle$, where x, x' are the unique two preimages of y [3].

Remarks. First we note that each image y appears with same probability (therefore,

[2]The functions should also be two-regular (each image has exactly two preimages), quantum safe (secure against quantum attackers) and collision resistant (hard to find two inputs with the same image).

[3]The uniqueness of the 2 preimages is due to the fact that the function is two-regular.

obtaining twice the same y happens with negligible probability). We now consider the first register $|x\rangle + |x'\rangle = |x_1 \cdots x_n\rangle + |x'_1 \cdots x'_n\rangle$, where the subscripts denote the different bits of the corresponding preimages x and x'. We rewrite this:

$$\left(\otimes_{i \in \bar{G}} |x_i\rangle \right) \otimes \left(\prod_{j \in G} X^{x_j} \right) \left(|0 \cdots 0\rangle_G + |1 \cdots 1\rangle_G \right)$$

where $\bar{G}$ is the set of bits positions where x, x' are identical, G is the set of bits positions where the preimages differ, while we have suitably changed the order of writing the qubits. It is now evident that the state at the end of Stage 1 is a tensor product of isolated $|0\rangle$ and $|1\rangle$ states, and a Greenberger-Horne-Zeilinger (GHZ) state with random X's applied. The crucial observation is that the connectivity (which qubit belongs to the GHZ and which doesn't) depends on the XOR of the two preimages $x \oplus x'$ and is computationally impossible to determine, with non-negligible advantage, without the trapdoor information t_k.

Stage 2: Squeezing. The client instructs the server to measure each qubit i (except the output) in a random basis $\{|0\rangle \pm e^{i\alpha_i \pi/4} |1\rangle\}$ and return back the measurement outcome b_i. The output qubit is of the form $|+_\theta\rangle = 1/\sqrt{2}(|0\rangle + e^{i\theta} |1\rangle)$, where:

$$\theta = \frac{\pi}{4}(-1)^{x_n} \sum_{i=1}^{n-1} (x_i - x'_i)(4b_i + \alpha_i) \bmod 8 \tag{4.1}$$

Intuitively, measuring qubits that are not connected has no effect to the output, while measuring qubits within the GHZ part, rotates the phase of the output qubit (by a $(-(1)^{x_i}\alpha_i + 4b_i)\pi/4$ angle). The above intuition shows that our $\mathsf{HBC-QFactory}$ protocol is correct, as fully proven in Theorem Theorem 4.3.1.

Security. The protocol is secure, if we can prove that the server (or other third parties) cannot guess (obtain noticeable advantage in guessing) the classical description of the state, i.e. the value of θ. We consider an honest-but-curious server which means that he essentially follows the protocol and the security reduces in proving that the server cannot use his classical information to obtain *any* advantage in guessing the classical description of the (honest) quantum output.

The server does not know the two preimages x, x' and needs to guess θ (which is a three-bit string) from the value of the image y. The key technical part of the security proof is showing a variant of the Goldreich-Levin theorem [GL89], that (informally) states that the predicate represented by the inner product of the preimage of a one-way

function with a random vector, taken modulo 2, is indistinguishable from a random bit. In our case, θ has a similar expression (4.1) as it can be expressed as the inner product between the XOR of two preimages and a random vector taken modulo 8. We prove in Theorem 4.4.4 that if a computationally bounded server could obtain non-trivial advantage in guessing θ, then he could also break the property of "second preimage resistance" which we requested[4] for our function f_k. To prove this theorem we first express each of the 3 bits of θ as a XOR between a Goldreich-Levin type of predicate and some extra functions. Each of these predicates, instead of having a preimage in the inner product, they have the (bitwise) XOR of the two preimages. We therefore show that guessing any of those predicates would break the (stronger) assumption of collision resistance, reaching a contradiction. Then, to connect the hardness of computing the bits of θ (each of the three predicates) with the hardness of computing θ, we use the theorem [VV85] to address the issue of possible correlations. The technical hardest part of Theorem 4.4.4 is on the one hand that we fix all but one variable in the expression of each predicate (bit of θ), with an extra cost that is an inverse polynomial probability and on the other hand that we then use a "disentangling" trick to express the bits as the XOR between a Goldreich-Levin predicate and an extra function (now independent of the other variables).

Using the property of θ mentioned above, we then prove the full security of the $\mathsf{HBC-QFactory}$ protocol by showing the game-based security holds through a reduction to the hardcore function property of θ in Theorem 4.4.3. More specifically, we show that if the server runs honestly the protocol, but keeps a record of the classical transcript of the protocol together with the measurement outcomes he performs, then it is hard for him to correlate this internal view of the protocol with the protocol output $(\theta, |+_\theta\rangle)$.

The function. Our protocol relies on using functions that have a number of properties (one-way, trapdoor, two-regular, collision resistant (see Remark 4.1.1)), quantum safe). *Any* function satisfying those conditions is suitable for our protocol. While in first thought some of these appear hard to satisfy jointly (e.g. two-regularity and collision resistance), we give two constructions that achieve those properties from simpler functions: one from injective, homomorphic trapdoor one-way function and one from bijective trapdoor one-way function. Both constructions define a new function that has

[4]We actually request the strongest collision-resistance property that implies the second preimage resistance.

domain extended by one bit, and the value of that bit "decides" whether one uses the initial basic function or not.

More specifically, for the *first construction*, let us denote the injective, homomorphic, trapdoor one-way function by g_k, with k the public description of the function and t_k the trapdoor - used for the inversion of the function g_k. Then, we pick at random an element x_0 from the domain of g_k. The public description k' of the new desired function f will be k along with $g_k(x_0)$ and the corresponding trapdoor $t_{k'}$ of f would be t_k along with x_0.
Then, the function f, which is evaluated by the server, is described as: $f_{k'}(x,c) = g_k(x) + c \cdot g_k(x_0)$, which due to the homomorphic property of g, can be rewritten as: $f_{k'}(x,c) = g_k(x + c \cdot x_0)$. Now, we can see the 2-regularity property of f as, since g_k is injective, f will always have exactly 2 preimages of the form: x and $x + x_0$, which can always be efficiently computed from the image of f using $t_{k'}$. The one-wayness and quantum-safety of f are then proved by reduction to the one-wayness, respectively quantum-safety of g and finally, we prove the collision-resistance of f, by reducing it to the one-wayness of g.
For the *second construction*, we denote by g, a bijective, trapdoor one-way function. Then, in order to construct f, we will basically use 2 such functions g: the public description k' of f will consist of k_0 and k_1 – the public descriptions of g_{k_0} and g_{k_1} and the trapdoor of $f_{k'}$ will consist of the pair $t_{k'} = (t_{k_0}, t_{k_1})$ – the trapdoors of g_{k_0} and g_{k_1}.
Then, the function f, evaluated by the server, is described as: $f_{k'}(x,c) = g_{k_c}(x)$. Now, we can see the 2-regularity property of f as, since g_k is bijective, every y from the image of f, will have 2 preimages, namely the unique preimage of g_{k_0} and the unique preimage of g_{k_1}, which can be both computed from y using $t_{k'}$. Then, in section A.2 we prove the one-wayness and quantum-safety of f by reduction to one-wayness and quantum-safety of g and finally, we prove the second preimage-resistance of f by reduction to the one-wayness of one of the 2 functions g.

We then give a real implementation of the required function f based on the first type of construction, starting from the injective trapdoor one-way function defined in [MP12], which is derived from the Learning-With-Errors problem : $g_K(s,e) = s^t K + e^t$, where $s \in \mathbb{Z}_q^n$ and $K \in \mathbb{Z}_q^{n \times m}$, so using the above notation, we have $x = (s,e)$ and $x_0 = (s_0, e_0)$. This function also seems to satisfy the homomorphic property with respect to addition modulo q: $g_K(s,e) + g_K(s_0,e_0) \bmod q = (s^t K + e^t + s_0^t K + e_0^t) \bmod q = g_K((s+s_0) \bmod q, e+e_0)$. Unfortunately, things are not so simple, because the domain of the error vector $e \in \mathbb{Z}^m$ is such that each component of e is bounded by some

value μ, in order for g to be injective and correctly inverted using the trapdoor. This implies that g is homomorphic as long as $e+e_0$ is also bounded (in infinite norm) by μ. In our case, this means that $e+e_0$ may not be small enough to lie within g's domain, so it may be possible to have only one preimage for some image y. To overpass these problems, we do the following:
When we are constructing the trapdoor for the function f, in particular when we are sampling $x_0 = (s_0, e_0)$ from the domain of g, we will in fact sample e_0 from a smaller set, such that when it will be added together with a random input e, the total noise vector will still be small enough to lie within the domain of g with some good probability.

What we prove is that as long as e_0 is sampled from a subset of the domain of g such that e_0 is now bounded by $\mu' = \frac{\mu}{m}$, we will get that with at least a constant probability, $e+e_0$ is inside the domain of g, or in other words that f is now 2-regular with at least a constant probability. What remains to be proven is that when e_0 is restricted to this smaller domain, $g_K(s_0, e_0)$ still cannot be inverted by an adversary. Therefore, as a final step we prove that there exists an explicit choice of parameters such that both g and the restriction of g to the domain of e_0 are one-way functions and such that all the other properties of g are preserved.

As a result, under this choice of parameters, we obtain our desired function f, satisfying all required properties: one-wayness, trapdoor, collision-resistance and 2-regularity (with at least constant probability), where the hardness of inverting f is proven by reduction to worst-case hardness of approximating short vectors problems, with polynomial approximation factor, which is the current standard in lattice-based cryptosystems.

Remark 4.1.1. *It appears that the second preimage resistance property will be enough to prove the security of our scheme in the honest-but-curious setting. However, as soon as the server can be malicious, the collision resistance property will be very important, else the server might forge known valid states, which would break the security.*

4.2 $CC-RSP_\theta$ Primitive

In many distributed protocols the required communication consists of sending sequence of single qubits prepared in random states that are unknown to the receiver (and any other third parties). What we want to achieve is a way to generate remotely single qubits that are random and (appear to be) unknown to all parties but the client that gives the instructions.

Definition 4.2.1. *Let* $|+_\theta\rangle = 1/\sqrt{2}\left(|0\rangle + e^{i\theta}|1\rangle\right)$. *We define the set of states:*

$$R := \{|+_\theta\rangle\} \quad \textit{where} \quad \theta \in \{0, \pi/4, \pi/2, \cdots, 7\pi/4\} \tag{4.2}$$

By including magic states ($\left|+_{\pi/4}\right\rangle$), this set of states can be viewed as a "universal" resource, as applying Clifford operations on those states is sufficient for universal quantum computation. Furthermore, it is sufficient to implement both Blind Quantum Computation (e.g.[BFK09]) and Verifiable Blind Quantum Computation (e.g.[FKD17]).

We emphasize that the aim of defining an ideal functionality is to highlight the task we want to achieve (in terms of correctness) with our protocol $\mathsf{HBC-QFactory}$, rather than using it in a simulation or composable security definition.

Protocol 1 Primitive: Clasical Channel Remote State Preparation ($\mathsf{CC-RSP}_\theta$)

Requirements: Client is a purely classical party with no access to quantum resources.

Public Information: A distribution on pairs of lists M, intuitively containing the values of the classical variables used by the client and by the server.

Trusted Party:

– With some probability p returns to both parties abort, otherwise:

– Samples $(m_C, m_S) \leftarrow M$

– Samples $\theta \leftarrow \{0,1\}^3 \cdot \frac{\pi}{4}$

– Prepares a qubit in state $|+_\theta\rangle$

Outputs:

– Either returns abort to both client and server

– Or returns (m_C, θ) to the client, and $(m_S, |+_\theta\rangle)$ to the server

---

Remark 4.2.2. *(i) The outcome of this primitive is the client "sending" the qubit* $|+_\theta\rangle$ *(that she knows) to the server, thus simulating a quantum channel. (ii) We note that there is an abort possibility and some auxiliary classical messages* (m_C, m_S)*, both included to make the primitive general enough to allow for our construction. Furthermore, the classical description of the qubit,* θ*, and the classical messages* (m_C, m_S) *are totally uncorrelated (as* θ *is chosen randomly for each* (m_C, m_S)*). (iii) While the server* can *learn something about the classical description (e.g. by measuring the qubit), this information is limited and is the exact same information that he could obtain if the client had prepared and send a random qubit.*

4.3 The Real Protocol

We assume the existence[5] of a family $\{f_k\colon \{0,1\}^n \to \{0,1\}^m\}_{k\in K}$ of trapdoor one-way functions that are two-regular and collision resistant (or the weaker second preimage resistance property, see Remark 4.1.1) even against a quantum adversary. For any y, we will denote by $x(y)$ and $x'(y)$ the two unique different preimages of y by f_k (if the y is clear, we may remove it from $x(y)$ and $x'(y)$ and denote the 2 preimages as x and x'). Note that because of the two-regularity property $m \geq n-1$. We use subscripts to denote the different bits of the strings.

Protocol 2 Real HBC – QFactory Protocol

Requirements:

Public: A family $\mathcal{F} = \{f_k : \{0,1\}^n \to \{0,1\}^m\}$ of trapdoor one-way functions that are quantum-safe, two-regular and collision resistant (or second preimage resistant, see Remark 4.1.1)

Input:

– Client: uniformly samples a set of random three-bits strings $\alpha = (\alpha_1, \cdots, \alpha_{n-1})$ where $\alpha_i \leftarrow \{0,1\}^3$, and runs the algorithm $(k, t_k) \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^n)$. The α and k are public inputs (known to both parties), while t_k is the "private" input of the Client.

Stage 1: Preimages superposition

– Client: instructs Server to prepare one register at $\otimes^n H|0\rangle$ and second register initiated at $|0\rangle^m$

– Client: returns k to Server and the Server applies U_{f_k} using the first register as control and the second as target

– Server: measures the second register in the computational basis, obtains the outcome y and returns this result y to the Client. Here, an honest Server would have a state $(|x\rangle + |x'\rangle) \otimes |y\rangle$ with $f_k(x) = f_k(x') = y$ and $y \in \mathrm{Im}\, f_k$.

Stage 2: Squeezing

– Client: instructs the Server to measure all the qubits (except the last one) of the first register in the $\{|0\rangle \pm e^{\alpha_i \pi/4}|1\rangle\}$ basis. Server obtains the outcomes $b = (b_1, \cdots, b_{n-1})$ and returns the result b to the Client.

– Client: using the trapdoor t_k computes x, x'. Then check if the n-th bit of x and x' (corresponding to the y received in stage 1) are the same or different. If they are the same, returns abort, otherwise, obtains the classical description of the Server's state.

Output: If the protocol is run honestly, when there is no abort, the state that Server has is $|+_\theta\rangle$, where the Client (only) knows the classical description (see Theorem 4.3.1):

$$\theta = \frac{\pi}{4}(-1)^{x_n} \sum_{i=1}^{n-1} (x_i - x_i')(4b_i + \alpha_i) \bmod 8 \tag{4.3}$$

[5] See section 4.5 for our function. With that choice, we are guaranteed that the last bits of the two preimages are always different, and thus no need for an abort. We keep the protocol general so that different functions can be used.

Remarks: The first thing to note is that the server should not only be unable to guess θ from his classical communications, but he should also be unable to distinguish it from a random string with probability greater than negligible. We will prove this later, but for now it is enough to point out that θ depends on the preimages x and x' of y (which the Client can obtain using t_k).

The second thing to note is that while our expression of θ resembles the inner product in the Goldreich-Levin (GL) theorem, it differs in a number of places and our proof (that θ is a hard-core function), while it builds on GL theorem proof, is considerably more complicated. Details can be found in the security proof, but here we simply mention the differences: (i) our case involves three-bits rather than a predicate, and the different bits, if we view them separately, may not be independent, (ii) we have a term $(x-x')$ rather than a single preimage, so rather than the one-way property of the function we will need the second preimage resistance and (iii) for the same reason, if we view our function as an inner product, it can take both negative and positive values ($(x-x')$ could be negative).

A third thing to note is that we have singled-out the last qubit of the first register, as the qubit that will be the output qubit. One could have a more general protocol where the output qubit is chosen randomly, or, for example, in the set of the qubits that are known to have different bit values between x and x', but this would not improve our analysis so we keep it like this for simplicity. Moreover, while the "inner product" normally involves the full string x that one tries to invert, in our case, it does not include one of the bits (the last) of the string we wish to invert. It is important to note, that it does not change anything to our proofs, since if one can invert all the string apart from one bit with inverse polynomial probability of success, then trivially one can invert the full string with inverse polynomial probability (by randomly guessing the remaining bit or by trying out both values of that bit). Therefore, all the proofs by contradiction are still valid and in the remaining, for notational simplicity, we will take the inner products to involve all n bits.

4.3.1 Correctness and intuition

Theorem 4.3.1. *If both the Client and the Server follow Protocol 2, the protocol aborts when $x_n = x'_n$, while otherwise the Server ends up with the output (single) qubit being in the state $|+_\theta\rangle$, where θ is given by Eq. (4.3).*

Proof. In the first stage, before the first measurement, but after the application of U_{f_k},

the state is $\sum_x |x\rangle \otimes |f_k(x)\rangle$. What the measurement does, is that it collapses the first register in the equal superposition of the two unique preimages of the measured $y = f_k(x) = f_k(x')$, in other words in the state $(|x\rangle + |x'\rangle) \otimes |y\rangle$. It is not possible, even for malicious adversary (not considered here), to force the output of the measurement to be a given y (see [Aar05] for relation of PostBQP with BQP). This completes the first stage of the protocol. Before proceeding with the proof of correctness we make three observations.

By the second preimage resistance property of the trapdoor function, learning x is not sufficient to learn x' but with negligible probability, and intuitively, by the stronger collision resistance property, even a malicious server cannot forge a state $|x\rangle + |x'\rangle$ (with $f(x) = f(x')$) fully known to him.

Then, we examine what happens if the last bit of x and x' are the same and see why the protocol aborts. In this case, in the first register, the last qubit is in product form with the remaining state, and therefore any further measurements in stage 2 do not affect it, leaving it in the state $|x_n\rangle$. Because of this, the output state is not of the form of Eq. (4.3), while including this states in the set of possible outputs would change considerably our analysis.

Finally, we should note that the resulting state is essentially a Greenberger-Horne-Zeilinger (GHZ) state [GHZ89]: let G be the set of bits positions where x and x' differ (which include n – output qubit), while $\bar{G}$ is the set where they are identical. The state is then (where we no longer keep the qubits in order, but group them depending on their belonging to G or $\bar{G}$):

$$\left(\otimes_{i\in\bar{G}} |x_i\rangle \right) \otimes \left(\otimes_{j\in G} |x_j\rangle + \otimes_{j\in G} |x_j \oplus 1\rangle \right) \tag{4.4}$$

This can be rewritten as (up to trivial re-normalization):

$$\left(\otimes_{i\in\bar{G}} |x_i\rangle \right) \otimes \left(\prod_{j\in G} X^{x_j} \right) \left(|0\cdots 0\rangle_G + |1\cdots 1\rangle_G \right) \tag{4.5}$$

It is now evident that the state at the end of Stage 1 is a tensor product of isolated $|0\rangle$ and $|1\rangle$ states, and a GHZ state with random X's applied.

The important thing to note, is that the set G, that determines which qubits are in the GHZ state and which qubits are not, is *not* known to the server (apart from the fact that the position of the output qubit belongs to G since otherwise the protocol aborts). Moreover, this set denotes the positions where x and x' differ, which is given by the XOR of the two preimages $x \oplus x' := (x_1 \oplus x'_1, \cdots, x_n \oplus x'_n)$. Because of second preimage resistance of the function, the server should not be able to invert and obtain

$x \oplus x'$ apart with negligible probability (without access to the trapdoor t_k). This in itself does not guarantee that the Server cannot learn *any* information about the XOR of the preimages, but we will see that the actual form of the state is such that being able to obtain information would lead to invert the full XOR and thus break the second preimage resistance.

Now let us continue towards Stage 2. Measuring a qubit (other than the last one) in $\bar{G}$ has no effect on the last qubit (since it is disentangled). When the qubit index is in G, then measuring it at angle $\alpha_i\pi/4$ gives a phase to the output qubit of the form $(-(-1)^{x_i}\alpha_i + 4b_i)\pi/4$ as one can easily check[6]. Therefore, adding all the phases leads to the output state being:

$$|+_\theta\rangle\,;\ \theta = \frac{\pi}{4}(-1)^{x_n}\left(\sum_{i\in G\setminus\{n\}}\big(-\alpha_i(-1)^{x_i}+4b_i\big)\right) \bmod 8 \tag{4.6}$$

Because θ is defined modulo 2π and $-4 = 4 \bmod 8$, we can express the output angle in a more symmetrical way:

$$\theta = \frac{\pi}{4}(-1)^{x_n}\left(\sum_{i=1}^{n-1}(x_i - x_i')\big(4b_i+\alpha_i\big)\right) \bmod 8 \tag{4.7}$$

Note that because the angles are defined modulo 2π, one can represent this angle as a 3-bits string $\tilde{B}$ (interpreted as an integer) such that $\theta := \tilde{B}\times\frac{\pi}{4}$ and eventually remove the $(-1)^{x_n}$ if needed by choosing the suitable convention in defining x and x'.

□

A final remark is that in an honest run of this protocol, the measurement outcomes b_i and y are uniformly chosen from $\{0,1\}$ and $\mathrm{Im}(f_k)$ respectively. This justifies why in the honest-but-curious model we can view the protocol as sampling randomly the different α, y, b's.

4.4 Security of $\mathsf{HBC-QFactory}$

Here we will prove the security of $\mathsf{HBC-QFactory}$ (Protocol 2) against Honest-But-Curious adversaries in the game-based security model. Before proceeding further, it is worth stressing that this security level has three-fold importance. Firstly, the Honest-But-Curious model concerns any application of $\mathsf{CC-RSP}_\theta$ that involves a protocol

[6]The $(-1)^{x_i}$-term arises because of the commutation of $X_i^{x_i}$ with the measurement angle, and the final $X_n^{x_n}$ gate gives an overall $(-1)^{x_n}$ to the angle of deviation

where the adversaries are third parties that have access to the classical communication and nothing else. In this case, we can safely assume that the quantum part of the protocol is followed honestly and we only require to prove that the third parties learn nothing about the classical description of the state from the classical public communication. Second case of interest is scenarios where the "server" does not intend to sabotage/corrupt the computation but may be interested to learn (for free) extra information. In such case, the protocol should be followed honestly, since any non-reversible deviation other than copying classical information could corrupt the computation. Finally, the Honest-But-Curious case, as in the classical setting, is a first step towards proving the full security against malicious adversaries.

4.4.1 Game-Based Security Definition

Definition 4.4.1. *In the following,* HBC − QFactory *is said to be* secure *if for all QPT adversaries $\mathcal{A}$ the following game is won with probability at most $\frac{1}{2} + \mathsf{negl}(n)$:*

Game G_{sec}

Challenger		**Adversary $\mathcal{A}$**
$c \leftarrow_{\$} \{0,1\}$		
$(\tilde{k}, \tilde{t_k}) \leftarrow_{\$} Gen_{\mathcal{F}}(1^n)$		
$\tilde{\alpha} \leftarrow_{\$} \{0,1\}^{3(n-1)}$		
$\tilde{x} \leftarrow_{\$} Dom(f_{\tilde{k}})$		
$\tilde{y} \leftarrow f_{\tilde{k}}(\tilde{x})$		
$\tilde{b} \leftarrow_{\$} \{0,1\}^{n-1}$		
$\theta^{(0)} \leftarrow Inv(\tilde{t_k}, \tilde{y}, \tilde{b})$		
$\theta^{(1)} \leftarrow_{\$} \{0,1\}^3 \cdot \frac{\pi}{4} - \{\theta^{(0)}\}$		
	$\xrightarrow{\tilde{k}, \tilde{\alpha}, \tilde{y}, \tilde{b}, \theta^{(c)}, \lvert +_{\theta^{(c)}} \rangle}$	
	$\xleftarrow{\tilde{c}}$	

Adversary $\mathcal{A}$ wins the game G_{sec} if and only if $c = \tilde{c}$.

Then what we want to show is that:

$$Pr[\mathcal{A} \text{ wins } G_{sec}] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{4.8}$$

In other words:

$$Pr[\mathcal{A}(\theta^{(c)}, |+_{\theta^{(c)}}\rangle) = c] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{4.9}$$

Remark 4.4.2. *Note that a stronger game could have been defined, where the* $|+_\theta\rangle$ *that is sent to the adversary is always* $|+_{\theta^{(0)}}\rangle$*, along with* $\theta^{(c)}$*. This would be closer to the actual view of an honest but curious adversary, but it is not possible to prove the security of this game, no matter how secure the underlying protocol is. Indeed, given* $\theta^{(c)}$ *and* $|+_{\theta^{(0)}}\rangle$*, it is always possible to measure the* $|+_{\theta^{(0)}}\rangle$ *in the basis* $\theta^{(c)}$ *and output* $\tilde{c} = 0$ *only if the measurement outcome was 0.*

The case $c = 0$ corresponds to the adversary receiving a transcript of the protocol $(\tilde{k}, \tilde{y}, \tilde{b})$ along with the real output of the protocol $(\theta^{(0)}, |+_{\theta^{(0)}}\rangle)$ matching to this transcript. The case $c = 1$ corresponds to adversary receiving a transcript of the protocol along with a random output of the protocol $(\theta^{(1)}, |+_{\theta^{(1)}}\rangle)$. The game ensures that an adversary cannot distinguish these two views. The security of G_{sec} tries to capture the following idea: If an adversary (server) runs honestly the protocol and keeps a record of the measurements he performs together with the transcripts he receives during the protocol, it must be hard for him to correlate this internal view with the output of the protocol $(\theta, |+_\theta\rangle)$. This is why in the game we ask the adversary to distinguish whether he has access to either a correlated or an uncorrelated θ.

4.4.2 Game-Based Security of $\mathsf{HBC-QFactory}$

Theorem 4.4.3. *For any QPT adversary* $\mathcal{A}$*, the game* G_{sec} *can be won with probability at most* $\geq \frac{1}{2} + \mathsf{negl}(n)$.

To prove that $\mathsf{HBC-QFactory}$ is secure according to Definition 4.4.1 we will rely on the following result, proven in Section 4.4.3:

Theorem 4.4.4. *The function* θ *expressed as:*

$$\theta = \frac{\pi}{4}\left(\sum_{i=1}^{n-1}(x_i - x_i')(4b_i + \alpha_i)\right) \bmod 8 \tag{4.10}$$

as was defined in Protocol 2, is a hard-core function with respect to f_k*.*
NB: here the collision resistance is not needed and is replaced by the weaker second preimage resistance property.

Proof of Theorem 4.4.3. The main idea would be to use a reduction to the hardcore property of the state description θ. We will assume that there exists an adversary $\mathcal{A}$ that can win G_{sec} with probability $\frac{1}{2}+p$ and we will construct an adversary $\mathcal{A}'$ that can break the hard-core function property of θ with probability $\frac{1}{8}+\frac{p}{4}$. This implies that if the game G_{sec} can be won with inverse polynomial probability, the same applies to the hard-core function property, and hence we reach a contradiction.

More specifically, let us assume that the adversary $\mathcal{A}$ can win with probability $\frac{1}{2}+p_0$ when $c=0$ and with probability $\frac{1}{2}+p_1$ when $c=1$. Then, we have:

$$Pr[\mathcal{A} \text{ wins } G_{sec}] = Pr[\tilde{c}=c] = Pr[\tilde{c}=0\,|\,c=0]\cdot\frac{1}{2} + Pr[\tilde{c}=1\,|\,c=1]\cdot\frac{1}{2}$$

So we have: $Pr[\tilde{c}=0\,|\,c=0]+Pr[\tilde{c}=1\,|\,c=1]=1+2p$.
By denoting $Pr[\tilde{c}=0\,|\,c=0]=\frac{1}{2}+p_0$ and $Pr[\tilde{c}=1\,|\,c=1]=\frac{1}{2}+p_1$, we obtain: $p_0+p_1=2p$.

Now we can define the following adversary $\mathcal{A}'$ that will attack the hardcore property of the state description using the adversary $\mathcal{A}$ for the game G_{sec}. Namely, $\mathcal{A}'$ will receive the tuple (y,b,k) - representing the messages that an adversary has during an honest run of HBC – QFactory and will try to determine the underlying state description θ.

$\mathcal{A}'(y,b,k,\alpha)$

1 : $\phi \leftarrow_{\$} \{0,1\}^3 \cdot \frac{\pi}{4}$
2 : Sends $(k,\alpha,y,b,\phi,|+_\phi\rangle)$ to $\mathcal{A}$
3 : $\tilde{c} \leftarrow \mathcal{A}(k,\alpha,y,b,\phi,|+_\phi\rangle)$
4 : **if** $(\tilde{c}==0)$ **then**
5 : $\quad\tilde{\theta}\leftarrow\phi$
6 : **elseif** $(\tilde{c}==1)$ **then**
7 : $\quad\tilde{\theta}\leftarrow_{\$}\{0,1\}^3\cdot\frac{\pi}{4}-\{\phi\}$
8 : $\quad$**return** $\tilde{\theta}$

Now we want to determine the probability that the output of the adversary $\mathcal{A}$ is equal to the "true" $\theta(y,b,k,\alpha)$ that is obtained by $Inv(t_k,y,b,\alpha)$.

We will consider separately two cases, namely: (i) $\phi=\theta(y,b,k,\alpha)$ and (ii) $\phi\neq\theta(y,b,k,\alpha)$. Since ϕ is chosen randomly from the eight possible angles, it is clear that

case (i) occurs with probability $1/8$ and case (ii) occurs with probability $7/8$.

(i) $\mathcal{A}$ receives the "true" state ($c = 0$), so to win the game he needs to return $\tilde{c} = 0$. By definition this happens with $\frac{1}{2} + p_0$, and in this case $\mathcal{A}'$ also wins (since he outputs the correct state). The overall probability that all this happens, i.e. that $\mathcal{A}'$ succeeds in this case, is $\frac{1}{8} \cdot \left(\frac{1}{2} + p_0\right)$.

(ii) $\mathcal{A}$ receives one of the "false" states ($c = 1$), and thus to win G_{sec} he needs to return $\tilde{c} = 1$. By definition this happens with probability $\frac{1}{2} + p_1$. Now, in this case, $\mathcal{A}'$ has essentially ruled-out one of the eight possible states. His random guess, after ruling-out one state, succeeds with probability $\frac{1}{7}$. Combining all this together we see that $\mathcal{A}'$ succeeds with probability $\frac{7}{8} \cdot \left(\frac{1}{2} + p_1\right) \cdot \frac{1}{7}$.

More explicitly, the probability that $\mathcal{A}'$ breaks the hardcore property of θ is (where we denoted the "true" $\theta(y,b,k,\alpha)$ by simply θ):

$$\begin{aligned}
&Pr[\mathcal{A}'(y,b,k,\alpha) = \theta] = Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi = \theta] \cdot Pr[\phi = \theta] + \\
&+ Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi \neq \theta] \cdot Pr[\phi \neq \theta] \\
&= \frac{1}{8} \cdot Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi = \theta] + \frac{7}{8} \cdot Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi \neq \theta] \\
&= \frac{1}{8} Pr[\tilde{c} = 0 \,|\, c = 0] + \frac{7}{8} Pr[\tilde{c} = 1 \text{ and random } \tilde{\theta} = \theta | c = 1] \\
&= \frac{1}{8}\left(\frac{1}{2} + p_0\right) + \frac{7}{8} \cdot \left(\frac{1}{2} + p_1\right) \cdot \frac{1}{7} = \frac{1}{8} + \frac{p_0 + p_1}{8} = \frac{1}{8} + \frac{p}{4}
\end{aligned} \tag{4.11}$$

We showed that Adversary $\mathcal{A}'$ succeeds with probability $\frac{1}{8} + \frac{p}{4}$ in guessing $\theta(y,b,k,\alpha)$, where p is the advantage $\mathcal{A}$ has in G_{sec}. However, we will prove in Theorem 4.4.4, that $\mathcal{A}$ can't win with probability better than $1/8 + \mathsf{negl}(n)$. Contradiction. □

4.4.3 Hardcore Function θ

Proof Sketch of Theorem 4.4.4. In Protocol 2, the adversary (Server) can only use the classical information that he possesses (k, y, α, b) in order to try and guess with some probability the value of θ in the case that there is no abort. Since the adversary follows the honest protocol, the choices of y, b are truly random (and not determined by the adversary as he could in the malicious case).

Outline of proof sketch: We first express the classical description of the state into expressions for each of the corresponding three bits. The aim is to prove that it is impossible to distinguish the sequence of these three bits from three random bits with

non-negligible probability. To show this we follow five steps. In **Step 1** we express each of the the bits as a sum mod two, of an inner product (of the form present in GL theorem) and some other terms. In **Step 2** we show that guessing the sum modulo two of the two preimages breaks the second preimage resistance of the function and thus is impossible. We assume that the adversary can achieve some inverse polynomial advantage in guessing certain predicates and in the remaining steps we show that in this case he can obtain a polynomial inversion algorithm for the one-way function f_k, and thus reach the contradiction. In **Step 3** we use the Vazirani-Vazirani Theorem 2.3.11 to reduce the proof of hard-core function to a number of single hard-core bits (predicates). In **Step 4** we use a result that allows us to fix all but one variable in each expression, with an extra cost that is an inverse polynomial probability and therefore the (fixed variables) guessing algorithm still needs to have negligible success probability. Finally, in **Step 5**, we reduce all the predicates in a form of a known hard-core predicate XOR with a function that involves variables not included in that predicate. Using the previous step, it reduces to guessing the XOR of a hard-core predicate with a constant, which is bounded by the probability of guessing the (known to be hard-core) predicate.

Here we give the sketch described above, while the full proof can be found in the section A.1. Let us start by defining:

$$\widetilde{B} = \widetilde{B}_1\widetilde{B}_2\widetilde{B}_3 = \left(\sum_{i=1}^{n-1}(x_i - x'_i)(4b_i + \alpha_i)\right) \bmod 8 \tag{4.12}$$

where $\widetilde{B}_i$ are single bits. Moreover, we treat x, x' as vectors in $\{0,1\}^n$; we define $\alpha^{(j)} = (\alpha_1^{(j)}, \cdots, \alpha_{n-1}^{(j)})$ the vector that involves the $j \in \{1,2,3\}$ bit of each of three-bit strings α, and we define $\tilde{x} := x \oplus x'$. We define z as a vector in $\{-1,0,1\}^n$ defined as the element-wise differences of the bits of x and x', i.e. $z_i = x_i - x'_i$. Finally, as in GL theorem, we will denote the inner product of 2 vectors a and b each with $n-1$ elements as: $\langle a, b\rangle = \sum_{i=1}^{n-1} a_i b_i$.

We will prove that any QPT adversary $\mathcal{A}$ having all the classical information that Server has (y, α, b), can guess $\widetilde{B}$ with at most negligible probability:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ \alpha\leftarrow\{0,1\}^{3n} \\ b\leftarrow\{0,1\}^n}} [\mathcal{A}(f(x), \alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, b) = \widetilde{B_1}\widetilde{B_2}\widetilde{B_3}] \leq \frac{1}{8} + \mathsf{negl}(n) \tag{4.13}$$

where for simplicity we denote the function f instead of f_k. This means that the adversary $\mathcal{A}$ cannot distinguish $\widetilde{B}$ from a random three-bit string with non-negligible

probability.

Step 1: We decompose Eq. (4.12) into three separate bits, and use the variable $\tilde{x}, z$ defined above.

$$\begin{aligned}\widetilde{B}_3 &= \langle \tilde{x}, \alpha^{(3)} \rangle \bmod 2 \\ \widetilde{B}_2 &= \langle \tilde{x}, \alpha^{(2)} \rangle \bmod 2 \oplus h_2(z, \alpha^{(3)}) \\ \widetilde{B}_1 &= \langle \tilde{x}, \alpha^{(1)} \rangle \bmod 2 \oplus h_1(z, \alpha^{(3)}, \alpha^{(2)}, b) \end{aligned} \tag{4.14}$$

where the derivation and exact expressions for the functions h_1, h_2 are given in section A.1. We notice from Eq. (4.14) that each bit includes a term of the form $\langle \tilde{x}, \alpha^{(i)} \rangle \bmod 2$ which on its own is a hard-core predicate following the GL theorem.

Step 2: By the second preimage resistance we have:

$$\begin{aligned}&\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x' \text{ such that } f(x) = f(x') \text{ and } x \neq x'] \leq \mathsf{negl}(n) \Rightarrow \\ &\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x' \oplus x = \tilde{x}] \leq \mathsf{negl}(n)\end{aligned} \tag{4.15}$$

For each bit $j \in \{1,2,3\}$, separately we assume that the adversary can achieve an advantage in guessing $\widetilde{B}_j$ which is $\frac{1}{2} + \varepsilon_j(n)$. Then, similarly to GL theorem, we prove that if this $\varepsilon_j(n)$ is inverse polynomial, this leads to contradiction with Eq. (4.15) since one can obtain an inverse-polynomial inversion algorithm for the one-way function f.

Step 3: While each bit includes terms that on its own it would make it hard-core predicate (as stated in Step 1), if we XOR the overall bit with other bits it could destroy this property. To proceed with the proof that $\widetilde{B}$ is hard-core function we use the Vazirani-Vazirani theorem which states that it suffices to show that individual bits as well as combinations of XOR's of individual bits are all hard-core predicates. In this way one evades the need to show explicitly that the guesses for different bits are not correlated. To proceed with the proof, we use a trick that "disentangles" the different variables.

Step 4: We would like to be able to fix one variable and vary only the remaining, while at the same time maintain some bound on the guessing probability.

The advantage $\varepsilon_j(n)$ that we assume the adversary has for guessing one bit (or a XOR) is calculated "on average" over all the random choices of $(\tilde{x}, \alpha^{(i)}, b)$. Using Lemma 4.4.5 we can fix one-by-one all but one variable (applying the lemma iteratively, see section A.1). With suitable choices, the cardinality of the set of values that satisfies all these conditions is $O(2^n \varepsilon_j(n))$ for each iteration. Unless $\varepsilon_j(n)$ is negligible, this size is an inverse polynomial fraction of all values. This suffices to reach the contradiction. The actual inversion probability that we will obtain is simply a product

of the extra cost of fixing the variables with the standard GL inversion probability. This extra cost is exactly the ratio between the cardinality of the Good sets (defined below) and the set of all values.

Lemma 4.4.5. *Let* $\Pr_{(v_1,\cdots,v_k)\leftarrow\{0,1\}^n\times\cdots\times\{0,1\}^n}[Guessing] \geq p+\varepsilon(n)$*, then for any variable* v_i*, there exists a set* $Good_{v_i} \subseteq \{0,1\}^n$ *of size at least* $\frac{\varepsilon(n)}{2}2^n$*, such that for all* $v_i \in Good_{v_i}$*, we have:*

$$\Pr_{(v_1,\cdots,\not{v_i},\cdots,v_k)\leftarrow\{0,1\}^n\times\cdots\times\{0,1\}^n}[Guessing] \geq p+\frac{\varepsilon(n)}{2}$$

where the latter probability is taken over all variables except v_i*.*

Step 5: If the expression we wish to guess involves XOR of terms that depend on different variables, then by using Step 4 we can fix the variables of all but one term. Then we note that trying to guess a bit (that depends on some variable and has expectation value close to $1/2$) is at least as hard as trying to guess the XOR of that bit with a constant. For example, if the bit we want to guess is $\langle\widetilde{x},r_1\rangle \bmod 2 \oplus h(z,r_2,r_3)]$ and we have a bound on the guessing probability where only r_1 is varied, then we have: [7]

$$\begin{aligned}&\Pr_{r_1\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r_1,r_2,r_3) = \langle\widetilde{x},r_1\rangle \bmod 2 \oplus h(z,r_2,r_3)] \leq \\ &\Pr_{r_1\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r_1,r_2,r_3) = \langle\widetilde{x},r_1\rangle \bmod 2]\end{aligned} \tag{4.16}$$

We note that all bits of $\widetilde{B}$ and their XOR's can be brought in this form. Then using this, we can now prove security, as the r.h.s. is exactly in the form where the GL theorem provides an inversion algorithm for the one-way function f. For details, see section A.1. □

4.5 Function Constructions

For our Protocol 2 we need a trapdoor one-way function that is also quantum-safe, two-regular and second preimage resistant (or the stronger collision resistance property). These properties may appear to be too strong to achieve, however, we give here methods to construct functions that achieve these properties starting from trapdoor one-way functions that have fewer (more realistic) conditions, and we specifically give one example that achieves *all* the desired properties. In particular we give:

[7] Here and in the full proof, when we compare winning probabilities for QPT adversaries, it is understood that we take the adversary that maximises these probabilities.

- A general construction given either (i) an injective, homomorphic (with respect to any operation[8]) trapdoor one-way function or (ii) a bijective trapdoor one-way function, to obtain a two-regular, second preimage resistant[9], trapdoor one-way function. In both cases the quantum-safe property is maintained (if the initial function has this property, so does the constructed function).
- (taken from [MP12]) A method of how to realise injective quantum-safe trapdoor functions derived from the LWE problem, that has certain homomorphic property.
- A way to use the first construction with the trapdoor from [MP12] that requires a number of modifications, including relaxation of the notion of two-regularity. The resulting function satisfy all the desired properties if a choice of parameters that satisfy multiple constraints, exists.
- A specific choice of these parameters, satisfying all constraints, that leads to a concrete function with all the desired properties.

4.5.1 Obtaining two-regular, collision resistant/second preimage resistant, trapdoor one-way functions

Here we give two constructions. The first uses as starting point an injective, homomorphic trapdoor function while the second a bijective trapdoor function. While we give both constructions, we focus on the first construction since (i) we can prove the stronger collision-resistance property and (ii) (to our knowledge) there is no known bijective trapdoor function that is believed to be quantum-safe.

Theorem 4.5.1. *If $\mathcal{G}$ is a family of injective, homomorphic, trapdoor one-way functions, then there exists a family $\mathcal{F}$ of two-regular, collision resistant, trapdoor one-way functions. Moreover the family $\mathcal{F}$ is quantum-safe if and only if the family $\mathcal{G}$ is quantum-safe.*

From now on, we consider that any function $g_k \in \mathcal{G}$ has domain D and range R and let $+_D$ be the closed operation on D and $+_R$ be the closed operation on R such that g_k is the morphism between D and R with respect to these 2 operations:

$$g_k(a) +_R g_k(b) = g_k(a +_D b) \ \ \forall a, b \in D$$

[8] in particular it is only required to be homomorphic once for this operation

[9] In (i) we prove the stronger collision-resistant property.

We also denote the operation $-_D$ on D, the inverse operation of $+_D$, specifically: $a +_D b^{-1} = a -_D b \ \forall a,b \in D$ and 0_D be the identity element for $+_D$.
Then, the family $\mathcal{F}$ is described by the following PPT algorithms:

$\texttt{FromInj.Gen}_{\mathcal{F}}(1^n)$

1 : $(k,t_k) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$ // k is an index of a function from $\mathcal{G}$ and t_k is its associated trapdoor
2 : $x_0 \leftarrow_{\$} D \setminus \{0_D\}$ // $x_0 \neq 0_D$ to ensure that the 2 preimages mapped to the same output are distinct
3 : $k' := (k, g_k(x_0))$ // the description of the new function
4 : $t'_k := (t_k, x_0)$ // the trapdoor associated with the function $f_{k'}$
5 : **return** k', t'_k

The Evaluation procedure receives as input an index k' of a function from $\mathcal{F}$ and an element $\bar{x}$ from the function's domain ($\bar{x} \in D \times \{0,1\}$):

$\texttt{FromInj.Eval}_{\mathcal{F}}(k', \bar{x})$

return $f_{k'}(\bar{x})$

where every function from $\mathcal{F}$ is defined as:

$$f_{k'} : D \times \{0,1\} \to R$$
$$f_{k'}(x,c) = \begin{cases} g_k(x), & \text{if } c = 0 \\ g_k(x) +_R g_k(x_0) = g_k(x +_D x_0)^{10}, & \text{if } c = 1 \end{cases}$$

$\texttt{FromInj.Inv}_{\mathcal{F}}(k', y, t'_k)$

1 : // y is an element from the image of $f_{k'}$, $k' = (k, g_k(x_0))$, $t'_k = (t_k, x_0)$
2 : $x_1 := Inv_{\mathcal{G}}(k, y, t_k)$
3 : $x_2 := x_1 -_D x_0$
4 : **return** $(x_1, 0)$ *and* $(x_2, 1)$ // the unique 2 preimages corresponding to
5 : // an element from the image of $f_{k'}$

Proof. To prove Theorem 4.5.1 we give below five lemmata showing that, the family $\mathcal{F}$ of functions defined above, satisfies the following properties: (i) two-regular, (ii) trapdoor, (iii) one-way, (iv) collision-resistant and (v) quantum-safe. □

Lemma 4.5.2 (two-regular)**.** *If $\mathcal{G}$ is a family of injective, homomorphic functions, then $\mathcal{F}$ is a family of two-regular functions.*

[10] The last equality follows since each function g_k from $\mathcal{G}$ is homomorphic

Proof. For every $y \in \operatorname{Im} f_{k'} \subseteq R$, where $k' = (k, g_k(x_0))$:

1. Since $\operatorname{Im} f_{k'} = \operatorname{Im} g_k$ and g_k is injective, there exists a unique $x := g_k^{-1}(y)$ such that $f_{k'}(x,0) = g_k(x) = y$.

2. Assume x' such that $f_{k'}(x',1) = y$. By definition $f_{k'}(x',1) = g_k(x' +_D x_0) = y$, but g_k is injective and $g_k(x) = y$ by assumption, therefore there exists a unique $x' = x -_D x_0$ such that $f_{k'}(x',1) = y$

Therefore, we conclude that:

$$\forall\, y \ \in \operatorname{Im} f_{k'} :\ f_{k'}^{-1}(y) := \{(g_k^{-1}(y),0), (g_k^{-1}(y) -_D x_0, 1)\} \tag{4.17}$$

□

Lemma 4.5.3 (trapdoor)**.** *If $\mathcal{G}$ is a family of injective, homomorphic, trapdoor functions, then $\mathcal{F}$ is a family of trapdoor functions.*

Proof. Let $y \in \operatorname{Im} f_{k'} \subseteq R$. We construct the following inversion algorithm:

$\mathit{Inv}_{\mathcal{F}}(k', y, t_k')$

1 : **//** $t_k' = (t_k, x_0),\ k' = (k, g_k(x_0))$

2 : $x := \mathit{Inv}_{\mathcal{G}}(k, y, t_k)$

3 : **return** $(x,0)$ and $(x -_D x_0, 1)$

□

Lemma 4.5.4 (one-way)**.** *If $\mathcal{G}$ is a family of injective, homomorphic, one-way functions, then $\mathcal{F}$ is a family of one-way functions.*

Proof. We prove it by contradiction. We assume that there exists a QPT adversary $\mathcal{A}$ that can invert a function in $\mathcal{F}$ with non-negligible probability P (i.e. given $y \in \operatorname{Im} f_{k'}$ to return a correct preimage of the form (x', b) with probability P). We then construct a QPT adversary $\mathcal{A}'$ that inverts a function in $\mathcal{G}$ with the same non-negligible probability P reaching a contradiction, since $\mathcal{G}$ is one-way by assumption.

From Eq. (4.17) of Lemma 4.5.2 we know the two preimages of y are: (i) $(g_k^{-1}(y), 0)$ and (ii) $(g_k^{-1}(y) -_D x_0, 1)$. We see that information on $g_k^{-1}(y)$ is obtained in both cases, i.e. obtaining any of these two preimages, is sufficient to recover $g_k^{-1}(y)$ if x_0 is known. We now construct an adversary $\mathcal{A}'$ that for any function $g_k : D \to R$, inverts any output $y = g_k(x)$ with the same probability P that $\mathcal{A}$ succeeds.

$\mathcal{A}'(k,y)$

1 : $x_0 \leftarrow_\$ D \setminus \{0_D\}$ **/** $\mathcal{A}'$ knows x_0, but is not given to $\mathcal{A}$
2 : $k' := (k, g_k(x_0))$
3 : $(x', b) \leftarrow \mathcal{A}(k', y)$
4 : **if** $((b == 0) \wedge (g_k(x') == y)$ **then**
5 : **/** equivalent to $\mathcal{A}$ succeeded in returning the first preimage
6 : **return** x'
7 : **elseif** $((b == 1) \wedge (g_k(x' +_D x_0)) == y)$ **then**
8 : **/** $\mathcal{A}$ succeeded in returning the second preimage
9 : **return** $x' +_D x_0$ **/** $\mathcal{A}'$ uses x_0 known from step 1
10 : **else /** $\mathcal{A}$ failed in giving any of the preimages (happens with probability $1 - P$)
11 : **return** 0

□

Lemma 4.5.5 (collision-resistance)**.** *If $\mathcal{G}$ is a family of injective, homomorphic, one-way functions, then any function $f \in \mathcal{F}$ is collision resistant.*

Proof. Assume there exists a QPT adversary $\mathcal{A}$ that given $k' = (k, g_k(x_0))$ can find a collision $(y, (x_1, b_1), (x_2, b_2))$ where $f_{k'}(x_1, b_1) = f_{k'}(x_2, b_2) = y$ with non-negligible probability P. From Eq. (4.17) we know that the two preimages are of the form $(x, 0), (x \triangle x_0, 1)$ where $g_k(x) = y$. It follows that when $\mathcal{A}$ is successful, by comparing the first arguments of the two preimages, can recover x_0.

We now construct a QPT adversary $\mathcal{A}'$ that inverts the function g_k with the same probability P, reaching a contradiction:

$\mathcal{A}'(k, g_k(x))$

1 : $k' := (k, g_k(x))$
2 : $(y, (x_1, b_1), (x_2, b_2)) \wedge x_1 \neq x_2 \leftarrow \mathcal{A}(k')$ **/** where y is an element from the image of $f_{k'}$
3 : **if** $f(x_1, b_1) == f(x_2, b_2) == y$
4 : **return** $x := x_1 -_D x_2$
5 : **else /** $\mathcal{A}$ failed to find collision of $f_{k'}$; happens with probability $(1 - P)$
6 : **return** 0

□

Lemma 4.5.6 (quantum-safe)**.** *If $\mathcal{G}$ is a family of quantum-safe trapdoor functions, with properties as above, then $\mathcal{F}$ is also a family of quantum-safe trapdoor functions.*

Proof. The properties that require to be quantum-safe is the one-wayness and collision resistance. Both these properties of $\mathcal{F}$ that we derived above were proved using reduction to the hardness (one-wayness) of $\mathcal{G}$. Therefore if $\mathcal{G}$ is quantum-safe, its one-wayness is also quantum-safe and thus both properties of $\mathcal{F}$ are also quantum-safe. □

Theorem 4.5.7. *If $\mathcal{G}$ is a family of bijective, trapdoor one-way functions, then there exists a family $\mathcal{F}$ of two-regular, second preimage resistant, trapdoor one-way functions. Moreover ,the family $\mathcal{F}$ is quantum-safe if and only if the family $\mathcal{G}$ is quantum-safe.*

The family $\mathcal{F}$ is described by the following PPT algorithms, where each function $g_k \in \mathcal{G}$ has domain D and range R:

$\texttt{FromBij.Gen}_{\mathcal{F}}(1^n)$

1: $(k_1, t_{k_1}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$
2: $(k_2, t_{k_2}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$
3: $k' := (k_1, k_2)$
4: $t'_k := (t_{k_1}, t_{k_2})$
5: **return** k', t'_k

$\texttt{FromBij.Eval}_{\mathcal{F}}(k', \bar{x})$

return $f_{k'}(\bar{x})$

where every function from $\mathcal{F}$ is defined as:

$$f_{k'} : D \times \{0,1\} \to R$$

$$f_{k'}(x,c) = \begin{cases} g_{k_1}(x), & \text{if } c = 0 \\ g_{k_2}(x), & \text{if } c = 1 \end{cases}$$

$\texttt{FromBij.Inv}_{\mathcal{F}}(k', y, t'_k)$

1 : / y is an element from the image of $f_{k'}$, $k' = (k_1, k_2)$, $t'_k = (t_{k_1}, t_{k_2})$
2 : $x_1 := Inv_{\mathcal{G}}(k_1, y, t_{k_1})$
3 : $x_2 := Inv_{\mathcal{G}}(k_2, y, t_{k_2})$
4 : **return** $(x_1, 0)$ *and* $(x_2, 1)$ / the unique 2 preimages corresponding to
5 : / an element from the image of $f_{k'}$

The proof of Theorem 4.5.7, using the family of function defined above, follows same steps as of Theorem 4.5.1 and is given in the section A.2.

4.5.2 Injective, homomorphic quantum-safe trapdoor one-way function based on LWE (from [MP12])

We outline the Micciancio and Peikert [MP12] construction of injective trapdoor one-way functions, naturally derived from the Learning-With-Errors problem. At the end we comment on the homomorphic property of the function, since this is crucial in order to use this function as the basis to obtain our desired two-regular, collision resistant trapdoor one-way functions.

The algorithm below generates the index of an injective function and its corresponding trapdoor. The matrix G used in this procedure, is a fixed matrix (whose exact form can be seen in [MP12]) for which the function from the family $\mathcal{G}$ with index G can be efficiently inverted without any trapdoor.

$\texttt{LWE.Gen}_{\mathcal{G}}(1^n)$

1 : $A' \leftarrow_{\$} \mathbb{Z}_q^{n \times \bar{m}}$
2 : $R \leftarrow_{\$} \mathcal{D}_{\alpha q}^{\bar{m} \times kn}$ / element-wise gaussian distribution with mean 0, standard deviation αq on $\bar{m} \times kn$ matrices
3 : $A := (A', G - A'R)$ / concatenation of matrices A' and $G - A'R$, representing the index of the function
4 : **return** (A, R)/ A - public function index, R - trapdoor

where the parameters k, α, q and $\bar{m}$ are defined in Theorem 4.5.11.

The actual description of the injective trapdoor function is given in the Evaluation algorithm below, where each function from $\mathcal{G}$ is defined on: $g_K : \mathbb{Z}_q^n \times L^m \to \mathbb{Z}_q^m$, and L is the domain of the errors in the LWE problem (the set of integers bounded in absolute value by the parameter μ):

$\underline{\texttt{LWE.Eval}_{\mathcal{G}}(K,(s,e))}$

1 : $y := g_K(s,e) = s^t K + e^t$

2 : **return** y

The inversion algorithm returns the unique preimage (s,e) corresponding to $b^t \in \mathrm{Im}(g_K)$. The algorithm uses as a subroutine the efficient algorithm Inv_G for inverting the function g_G, with G the fixed matrix mentioned before.

$\underline{\texttt{LWE.Inv}_{\mathcal{G}}(K,t_K,b^t)}$

1 : $b'^t := b^t \begin{bmatrix} R \\ I \end{bmatrix}$

2 : $(s',e') := Inv_G(b')$

3 : $s := s'$

4 : $e := b - K^t s$

5 : **return** s,e

We examine now whether the functions g_K are homomorphic with respect to some operation.
We first define the domain and the range as $D := \mathbb{Z}_q^n \times L^m$ and $R := \mathbb{Z}_q^m$. Then, given $a = (s_1,e_1) \in \mathbb{Z}_q^n \times L^m$ and $b = (s_2,e_2) \in \mathbb{Z}_q^n \times L^m$, the operation $+_D$ is defined as:

$$(s_1,e_1) +_D (s_2,e_2) = (s_1 + s_2 \bmod q, e_1 + e_2)$$

Given $y_1 = g_K(a) \in \mathbb{Z}_q^m$ and $y_2 = g_K(b) \in \mathbb{Z}_q^m$, the operation $+_R$ is defined as:

$$y_1 +_R y_2 = y_1 + y_2 \bmod q$$

Then, we can easily verify that:

$$g_K(s_1,e_1) + g_K(s_2,e_2) \bmod q = {s_1}^t K + {e_1}^t + {s_2}^t K + {e_2}^t \bmod q =$$
$$(s_1 + s_2 \bmod q)^t K + (e_1 + e_2)^t = g_K((s_1 + s_2) \bmod q, e_1 + e_2)$$

However, the sum of two error terms, each being bounded by μ, may not be bounded by μ. This means that the function is not (properly) homomorphic. Instead, what we conclude is that as long as the vector $e_1 + e_2$ lies inside the domain of g_K, then g_K is homomorphic. To address this issue, we will need to define a weaker notion of 2-regularity, and a (slight) modification of the `FromInj` construction to provide a desired function starting from the trapdoor function of [MP12].

4.5.3 A suitable δ-2 regular trapdoor function

Using the injective trapdoor function of Micciancio and Peikert [MP12] and the construction defined in the proof of Theorem 4.5.1, we derive a family $\mathcal{F}$ of collision-resistant trapdoor one-way function, but with a weaker notion of 2-regularity, called δ-2 regularity:

Definition 4.5.8 (δ-2 regular)**.** *A family of functions* $(f_k)_{k \leftarrow Gen_{\mathcal{F}}}$ *is said to be* δ*-2 regular, with* $\delta \in [0,1]$ *if:*

$$\Pr_{k \leftarrow Gen_{\mathcal{F}}, y \in \mathrm{Im}(f_k)} [\, |f_k^{-1}(y)| = 2 \,] \geq \delta$$

Given this definition, we should note here that in Protocol 2 we need to modify the abort case to include the possibility that the image y obtained from the measurement does not have two preimages (something that happens with at most probability $(1-\delta)$).

Theorem 4.5.9 (Existence of a δ-2 regular trapdoor function family)**.** *There exists a family of functions that are* δ*-2 regular (with* δ *at least as big as a fixed constant), trapdoor, one-way, collision resistant and quantum-safe, assuming that there is no quantum algorithm that can efficiently solve* SIVP_γ *for* $\gamma = \mathsf{poly}(n)$.

Proof. To prove this theorem, we define a function similar to the one in the `FromInj` construction, where the starting point is the function defined in [MP12]. Crucial for the security is a choice of parameters that satisfy a number of conditions given by Theorem 4.5.11 and proven in section A.3. The proof is then completed by providing a choice of parameters given in Lemma 4.5.12 that satisfies all conditions as it is shown in section A.4. □

Definition 4.5.10. *For a given set of parameter* P *chosen as in Theorem 4.5.11, we define the following functions, that are similar to the construction* `FromInj`*, with the difference that the key generation requires the trapdoor error to be sampled from a smaller subdomain (*$\alpha' < \alpha$*):*

$\texttt{REG2.Gen}_{\mathsf{P}}(1^n)$

1: $(A,R) \leftarrow \texttt{LWE.Gen}_{\mathcal{G}}(1^n)$
2: $s_0 \leftarrow \mathbb{Z}_q^{n,1}$
3: $e_0 \leftarrow \mathcal{D}_{\alpha' q}^{m,1}$ / $\alpha' < \alpha$
4: $b_0 := \texttt{LWE.Eval}(A,(s_0,e_0))$
5: $k := (A,b_0)$
6: $t_k := (R,(s_0,e_0))$
7: **return** (k,t_k)

$\texttt{REG2.Eval}_{\mathsf{P}}((A,b_0),(s,e,c))$

1: / *s is a random element in $\mathbb{Z}_q^{n,1}$, $c \in \{0,1\}$*
2: / *e is sampled uniformly and such that*
3: / *each component is smaller than μ*
4: **return** $\texttt{LWE.Eval}(A,(s,e)) + c \cdot b_0$

$\texttt{REG2.Inv}_{\mathsf{P}}((A,R,(s_0,e_0)),b)$

1: $(s_1,e_1) := \texttt{LWE.Inv}(R,b)$
2: **if** $||e_1 - e_0||_\infty \leq \mu$ **then return** $\perp$
3: **return** $((s_1,e_1,0),(s_1 - s_0 \bmod q, e_1 - e_0, 1))$

Note, that the pairs (s,e) and (s_0,e_0) correspond to x and x_0 of the `FromInj` construction of subsection 4.5.1. The idea behind this construction is that the noise of the trapdoor, e_0, is sampled from a set which is small compared to the noise of the input function. That way, when we will add the trapdoor (s_0,e_0) together with an input (s,e), the total noise will still be small enough to lie in the set of possible input noise with good probability, mimicking the homomorphic property needed in Theorem 4.5.1. Note that the parameters need to be carefully chosen, and a trade-off between probability of success and security exists.

We first introduce the following notation: for all $n,q,\mu \in \mathbb{Z}, \mu' \in \mathbb{R}$, let us define:

- $k := \lceil log(q) \rceil$
- $\bar{m} = 2n$
- $\omega = nk$
- $m := \bar{m} + \omega = 2n + nk$
- $\alpha' = \frac{\mu'}{\sqrt{m}q}$
- $\alpha = m\alpha'$
- C the constant in Lemma 2.9 of [MP12] which is around $\frac{1}{\sqrt{2\pi}}$
- $B = 2$ if q is a power of 2, and $B = \sqrt{5}$ otherwise.

Theorem 4.5.11 (Requirements on the parameters)**.** *If for all security parameters n (dimension of the lattice), there exist q (the modulus of* LWE*) and μ (the maximum amplitude of the components of the errors) such that:*

1. *m is such that $n = o(m)$ (required for the injectivity of the function (see e.g. [Vai]))*

2. $0 < \alpha < 1$

3. $\mu' = O(\mu/m)$ *(required to have constant probability to have two preimages)*

4. $\alpha' q \geq 2\sqrt{n}$ *(required for the* LWE *to* SIVP *reduction)*

5. $\frac{n}{\alpha'}$ *is* $\mathsf{poly}(n)$ *(representing, up to a constant factor, the approximation factor γ in the* SIVP_γ *problem) - for the standard hardness of the* SIVP *problem.*

6. $$\sqrt{m}\mu < \underbrace{\frac{q}{2B\sqrt{\left(C\cdot(\alpha\cdot q)\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1}}}_{r_{max}} - \mu'\sqrt{m}$$

   (required for the correctness of the inversion algorithm - r_{max} represents the maximum length of an error vector that one can correct using the [MP12] function[11], and the last term is needed in the proof of collision resistance to ensure injectivity even when we add the trapdoor noise, as illustrated in Figure 4.1)

then the family of functions of Definition 4.5.10 is δ-2 regular (with δ at least as big as a fixed constant), trapdoor, one-way and collision resistant (all these properties hold even against a quantum attacker), assuming that there is no quantum algorithm that can efficiently solve $SIVP_\gamma$ for $\gamma = \mathsf{poly}(n)$.

Proof. The proof follows by showing that the function with these constraints on the parameters is: (i) δ-2 regular, (ii) collision resistant, (iii) one-way and (iv) trapdoor. In section A.3 we give and prove one lemma for each of those properties. For an intuition of the choice of parameters see also Figure 4.1. □

[11] We chose to use the computational definition of [MP12], but this theorem can be easily extended to other definitions of the same paper, or even to other construction of trapdoor short basis)

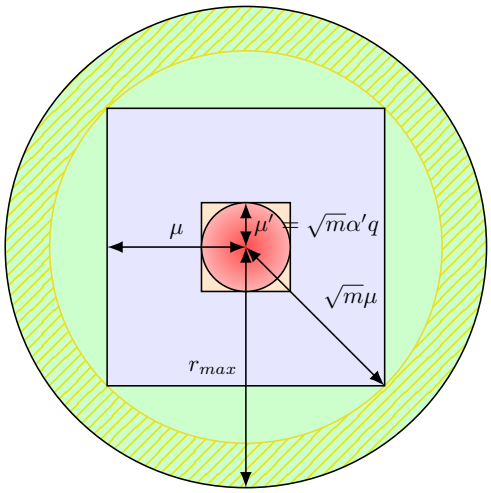


Figure 4.1: The red circle represents the domain of the error term from the trapdoor information, which is being sampled from a Gaussian distribution. The orange square is an approximation of this domain, which must satisfy that its length is much smaller (by a factor of at least m – the dimension of the error) than the length of the blue square, used for the actual sampling from the domain of the error terms, for which it is known that the trapdoor function is invertible, domain represented by the green circle (including the dashed part). The dashed part, representing the distance between the maximum domain for which the function is invertible (green circle) and the actual domain used for sampling (blue square) is needed to ensure that if there is a collision (x_1, x_2), then $x_1 = x_2 \pm x_0$.

4.5.4 Parameter Choices

Lemma 4.5.12 (Existence of parameters)**.** *The following set of parameters fulfills Theorem 4.5.11.*

$$
\begin{aligned}
n &= \lambda \\
k &= 5\lceil log(n) \rceil + 21 \\
q &= 2^k \\
\bar{m} &= 2n \\
\omega &= nk \\
m &= \bar{m} + \omega \\
\mu &= \left\lceil 2mn\sqrt{2+k} \right\rceil \\
\mu' &= \mu/m \\
B &= 2
\end{aligned}
$$

and α, α', C *are defined like in Theorem 4.5.11.*

The proof is given in section A.4. As a final remark, we stress that other choices of the parameters are possible (considering the trade-off between security and probability of success) and we have not attempted to find an optimal set.

4.6 Implementation of HBC − QFactory on IBM Quantum Cloud

Finally, in this section we provide the first proof-of-principle demonstration of a classical remote state preparation protocol. Very importantly, this implementation is not secure against a quantum adversary. The reason is that, as our 2-regular trapdoor function needs to be hard to invert this requires the size of the function to be sufficiently large, but at the same time we need to implement on server's quantum computer the unitary corresponding to this function. Therefore, given the current limited number of available qubits, we choose a 2-regular function that can be easily inverted by brute-force attacks. The scope of this implementation is to demonstrate the correctness and the randomness of the HBC − QFactory protocol.

For the implementation we use the IBM Quantum Experience service and will run all the experiments to prove the correctness and randomness of the protocol using both the simulator "`ibmq_qasm_simulator`" (32 available qubits) and the IBM real devices: "`ibmq_athens`" (5 available qubits) and "`ibmq_melbourne`" (16 available qubits).

4.6.1 Function Construction for Simulation

We will construct a specific 2-regular function (which given the limitations of the number of available qubits cannot be LWE-based or one-way). More specifically, we define the following 2-regular family of functions $f_{A,B} : \{0,1\}^3 \to \{0,1\}^2$, where the public key is a pair of 2 matrices $A, B \in \{0,1\}^{3\times 3}$:

$$f_{A,B}(x_1x_2x_3) = \left(\bigoplus_{i,j} A_{i,j}x_ix_j\right) \| \left(\bigoplus_{i,j} B_{i,j}x_ix_j\right) \tag{4.18}$$

where by $\|$ we denote the concatenation of the 2 bits.

To construct the **Key Generation Algorithm**, we will proceed as follows. We first sample uniformly at random the trapdoor information, $(d_0, e) \in \{0,1\}^2$, and we will construct the public key (A,B) as a function of the trapdoor in the following way:

$$A_{e+1,e+1} = A_{2-e,3} = B_{3,3} = 1, B_{2-e,2-e} = d_0, \text{ and all others elements are } 0 \tag{4.19}$$

For the **Inversion Algorithm**, we proceed in the following way:
Given a function image $y = (y_1, y_2) \in \{0,1\}^2$, we compute its 2 preimages $x = (x_1, x_2, x_3) \in \{0,1\}^3$ and $x' = (x'_1, x'_2, x'_3) \in \{0,1\}^3$ as:

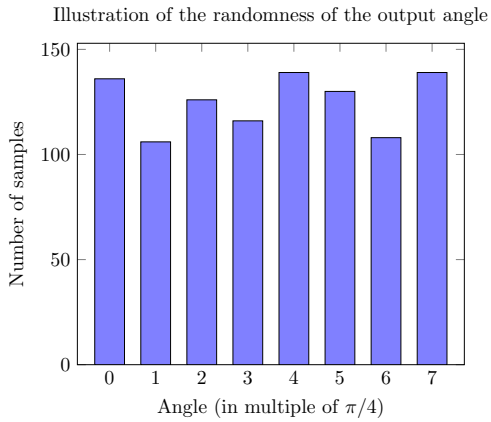
$$\begin{aligned} x_{2-e} &= 0, x_{1+e} = y_1, x_3 = y_2 \\ x'_{2-e} &= 1, x'_{1+e} = y_1 \oplus y_2 \oplus d_0, x'_3 = y_2 \oplus d_0 \end{aligned} \tag{4.20}$$

We emphasize that by constructing f in this particular way, we have the structure required for HBC – QFactory. Moreover, the circuit to implement the unitary $U_{f_{A,B}}$, it is simple to derive from A and B as we just need to implement a Toffoli gate (or CNOT if $i = j$) between i-th input qubit, j-th input qubit and the first output qubit whenever $A_{ij} = 1$, and similarly with the second output qubit whenever $B_{ij} = 1$.

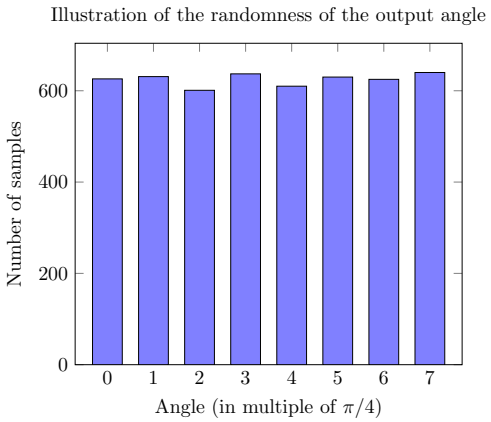
4.6.2 Randomness Results

We first want to indicate the *randomness* of the output of the HBC – QFactory protocol. By this we mean that we want to show that the quantum output is a $|+_\theta\rangle$ state, where θ is uniformly sampled from a set of 8 possible values: $\{0, \frac{\pi}{8}, \cdots, \frac{7\pi}{8}\}$.
The following plot indicates the distribution of θ, can be seen in Fig. 4.2a.

We run the protocol 1000 number of times, where the distribution is taken over the randomness of the 2 measurements: y (image register) and b (the final measurement - the preimage register) and over the uniform distribution of the measurement angles α.
The same protocol, run 5000 times shows an even closer to uniform distribution of θ, as seen in Fig. 4.2b.



(a) 1000 runs of HBC – QFactory using function $f_{A,B}$ described in Section 4.6.1. Plot depicts resulting output distribution.



(b) 5000 runs of HBC – QFactory

4.6.3 Correctness Results

For the *correctness* of the protocol we want to show 2-fold:

- The Server always obtains a $|+_\theta\rangle$ type of state on his side;
- The Client, by knowing the preimages of each image y, can always efficiently compute this θ;

Additionally, we will actually run 2 different types of experiments: using the simulator and a real quantum device. We proceed in the following two different manners.

When running the simulator we check that the state description the Client obtained and the quantum state resulted on the Server's side correspond to the same value of θ, in the following manner: For Client we run the algorithm described in HBC – QFactory and we compute the value θ. And now to check that Server obtained exactly the state $|+_\theta\rangle$, the Simulator allows us to get the description of the corresponding final quantum output state (as a vector), and we are therefore able to check that it matches perfectly. Therefore, the first correctness evidence is showed in Fig. 4.3, which depicts the values of θ_A obtained by Client and the the values of θ_B , where $|+_{\theta_B}\rangle$ is obtained by Server. As observed $\theta_A = \theta_B$ for all the runs of our protocol.

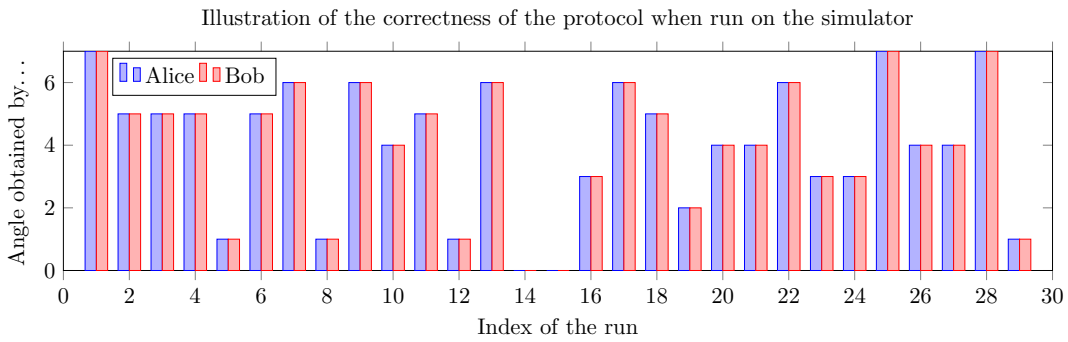


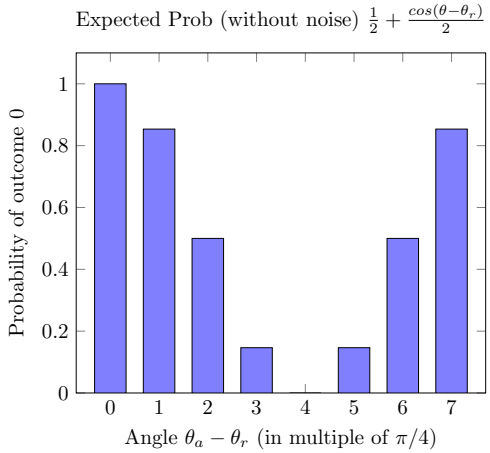
Figure 4.3: Values of θ_A (written as multiples of $\frac{\pi}{4}$) obtained by Client and the corresponding θ_B obtained by Server for 29 different runs of HBC – QFactory on the simulator, illustrating the correctness of the protocol.

For the run on the actual quantum device, Client obtains the value of θ in the same way. However, we are no longer able to get the vector description of the quantum state of Server, because now it corresponds to an actual physical qubit. Now, the only way to see what state Server has on his side is to perform a final measurement on the output qubit of the Server, which we will denote by $|\psi_{out}\rangle$.

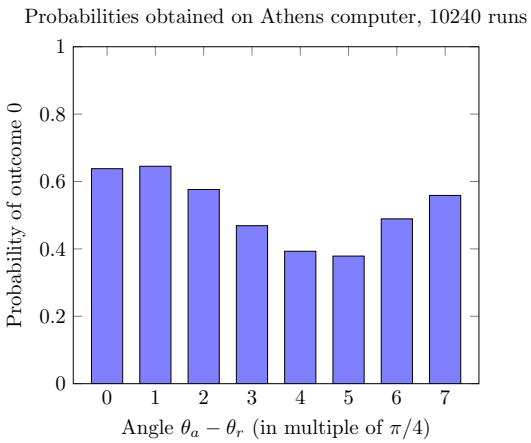
One way would be to measure the output state $|\psi_{out}\rangle$ in the $\{|+_\theta\rangle, |-_\theta\rangle\}$ basis, where θ would be the angle obtained by Client through her classical computations, and in this way for correctness we should have always outcome 1 for this measurement. Unfortunately, this is not possible, because the IBM Quantum Experience framework does not allow for intermediate measurements. Therefore, the way we will proceed for this is as follows. At the beginning of the protocol we pick a random θ_r and then we measure the final state of the Server $|\psi_{out}\rangle$ in the basis $\{|+_{\theta_r}\rangle, |-_{\theta_r}\rangle\}$. Then, once we receive all the measurement outcomes (preimage, image registers and $|\psi_{out}\rangle$) we first compute (using Client's algorithm) the actual θ and then we check:

- If $\theta = \theta_r$, we should obtain measurement outcome 0 with probability 1;
- If $\theta = \theta_r \pm \pi$ we should obtain 1 with probability 1;
- if $\theta = \theta_r \pm \frac{\pi}{2}$ we should get 1 with probability $\frac{1}{2}$;
- if $\theta = \theta_r \pm \frac{\pi}{4}$ we should get 0 with probability $\frac{1}{2} + \frac{\sqrt{2}}{4}$;
- In general the probability of the outcome 0 is equal to: $p(0) = \frac{1}{2} + \frac{cos(\theta - \theta_r)}{2}$

Therefore, in Fig.4.4a, we can see the expected probability of the measurement outcome to be 0 - which corresponds to the "ideal" quantum computer case. Then in Fig.4.4b we have the actual probability of outcome 0, as a result of running it on the "noisy" real quantum device - which corresponds to the real run.



(a) Ideal Case: θ is the honest angle expected by Client, θ_r is the random angle used to measure the state obtained by Server.



(b) Real Case: θ is the honest angle expected by Client, and θ_r is the random angle used to measure the state obtained by Server.

Chapter 5

QFactory against Malicious Server

In chapter 4, we described the primitive of classical-client remote state preparation as a method to enable fully classical parties to participate in secure quantum computation and communications protocols at the cost of relying on post-quantum computational security. The idea is to replace the quantum channel (used in many different protocols implementing blind or verifiable quantum computation) with a module running between a classical client and a quantum server. To achieve this, a classical party (the Client) is instructing a quantum party (the Server) to generate a qubit on the Server's side that is random and unknown to the Server, but known to the Client. Such a task can be accomplished under computational assumptions, but implausible to achieve with information-theoretic security as indicated by our results in chapter 3. In the previous chapter we showed how a classical client could implement this module (referred to as $\mathsf{HBC-QFactory}$) in order to achieve secure delegated universal quantum computing, but potentially also, other functionalities such as multi-party quantum computation.
While the $\mathsf{HBC-QFactory}$ Protocol guaranteed the secure preparation of $\{|+_\theta\rangle\}_{\theta\in\{0,\cdots,\frac{7\pi}{4}\}}$, in this contribution, we define a basic primitive consisting of BB84 states.
But more importantly, the security proof of $\mathsf{HBC-QFactory}$ was shown in a weak Honest-but-Curious model. In this work we extend the security proof, by giving a classical-client remote state preparation protocol that is secure against the strongest possible adversary (an arbitrarily deviating malicious Server). All our proofs are made using reductions to hardness assumptions (namely the Learning-With-Errors problem), and the simplicity of the main protocol suggests that an extension to a composable framework such as Constructive Cryptography [MR11] should be possible to analyze.
Following the modularity of $\mathsf{HBC-QFactory}$ we present in this chapter a universal, yet minimal functionality module that is fully secure, and can be used as a black box in

other Client-Server applications to replace the need for a reliable long-distance quantum communication network. The price one has to pay is a reduction from information-theoretic security (achievable using quantum communication) to post-quantum computational security via our modules.

In Chapter 4 we defined the primitive of classical-client remote state preparation ($\mathsf{CC}-\mathsf{RSP}$) that can replace the need for quantum channel between parties in certain quantum communication protocols, with the only trade-off being that the protocols would become computationally secure (against *quantum* adversaries). However, the proof of security was done in a weak model called "honest-but-curious".
In this chapter **our contributions** can be summarized as follows:

1. We present a new protocol called Malicious 4-states QFactory in section 5.2 that implements the $\mathsf{CC}-\mathsf{RSP}$ primitive for the generation of the quantum states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ (known as the BB84 states), given 2 cryptographic functions: 1) a trapdoor one-way function that is quantum-safe, two-regular and collision resistant and 2) a homomorphic, hardcore predicate. The novelty of this new protocol reflects in both simplicity of construction and proof, as well as enhanced security, namely the protocol is secure against any arbitrarily deviating adversary. The target output qubit set is one of the four BB84 states, states that form the core requirement of any quantum communication protocol.
   Then, in subsection 5.2.3, we present the security of the Malicious 4-states QFactory against any fully malicious server, by proving that the basis of the generated qubits are completely hidden from any adversary, using the properties of the two functions, the security being based on the hardness of the Learning-With-Errors problem.

2. While the above-mentioned results do not depend on the specific function used, the existence of such functions (with all desired properties) makes the functionality a practical primitive that can be employed as described in this paper. In section 5.3, we describe how to construct the two-regular, collision resistant, trapdoor one-way family of functions and the homomorphic, hardcore predicate. Furthermore, we prove using reductions in subsection 5.3.2 that the resulting functions maintain all the required properties.

3. In order to demonstrate the modular construction of the basic Malicious 4-states QFactory, we also present in section 5.4, a secure and efficient extension to the

functionality of generating 8 states, called the Malicious 8-states QFactory protocol (where the security refers to the fact that the basis of the new state is completely hidden). The set of output states $\{|+_\theta\rangle \,|\, \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\}$ (no longer within the Clifford group) are used in various protocols, including protocols for verifiable blind quantum computation.

4. While the protocol introduced in section 5.2 requires (for the security proof) a family of functions having 2 preimages with probability super-polynomially close to 1, we also define in section 5.5 a protocol named Malicious-Abort 4-states QFactory, that is secure when the functions have 2 preimages with only a constant (greater than $1/2$) probability. Indeed, even if the parameters used for the first category of functions are implicitly used in some protocols [Mah18], the second category of functions is strictly more secure and more standard in the cryptographic literature [Bra18]. The Malicious-Abort 4-states QFactory protocol is proven secure also for this second category of functions, assuming that the classical Yao's XOR lemma also applies for one-round protocols (with classical messages) against quantum adversaries.

5.1 Overview of Protocols and Proof Techniques

The Protocol. The general idea is that a classical client communicates with a quantum server instructing him to perform certain actions. By the end of the interaction, the client obtains a random value $B = B_1B_2 \in \{00, 01, 10, 11\}$, while the server (if he followed the protocol) ends up with the state $H^{B_1}X^{B_2}|0\rangle$, i.e. with one of the BB84 states. Moreover, the server, irrespective of whether he followed the protocol or how he deviated, cannot guess the value of the (basis) bit B_1 any better than making a random guess (more details in subsection 5.2.3).

This module is sufficient to perform (either directly or with simple extensions) multiple secure computation protocols including blind quantum computation. To achieve such a task, we require three central elements. Firstly, the quantum operations performed by the server should not be repeatable, in order to avoid letting the (adversarial) server run multiple times these operations and obtain multiple copies of the same output state. That would (obviously) compromise the security since direct tomography of a single qubit is straightforward. This can be achieved if the protocol includes a measurement of many qubits, where the probability of getting twice the same outcome

would be exponentially small. The second element is that the server should not be able to efficiently classically simulate the quantum computation that he needs to perform. This is to stop the server from running everything classically and obtaining the explicit classical description of the output state. This is achieved using techniques from post-quantum cryptography and specifically the Learning-With-Errors problem. Lastly, the computation has to be easy to perform for the client, since she needs to know the output state. This asymmetry (easy for client and hard for server) can be achieved in the computational setting, where the client has some extra trapdoor information. The protocol requires the following cryptographic primitives defined formally in Definition 5.2.1:

- $\mathcal{F}$: a family of 2-regular, collision resistant, trapdoor one-way functions (that can be constructed from a family of injective, homomorphic, trapdoor one-way functions $\mathcal{G}$);

- $d_0(t_k)$: a hardcore predicate of the index of the functions in $\mathcal{F}$. More precisely, every function $f_k \in \mathcal{F}$ has an associated hardcore bit d_0 that is hard to guess given only k, but easy to compute given the trapdoor t_k;

- h: a predicate such that $h(x) \oplus h(x') = d_0$ for any x, x' with $f_k(x) = f_k(x')$

Given these functions, the protocol steps are as follows.
The client sends the descriptions of the functions f_k (from the family $\mathcal{F}$) and h. The server's actions are described by the circuit given in Figure 5.1 (see section 5.2), classically instructed by the client: prepares one register at $\otimes^n H|0\rangle$ and second register at $|0\rangle^m$; then applies U_{f_k} using the first register as control and the second as target; measures the second register in the computational basis, obtains the outcome y. Through these steps server produces a superposition of the 2 preimages x and x' of y for the function f_k, i.e. $|x\rangle + |x'\rangle$. Next, server is instructed to apply the unitary corresponding to function h (targeting a new qubit $|0\rangle$) and to measure all but this new qubit in the Hadamard basis (the measurement outcomes will be denoted as b), which will be the output of the protocol. This last step intuitively magnifies the randomness of all the qubits to this final output qubit.
Essentially, this differs from the construction of $\mathsf{HBC-QFactory}$ in 2 points: the use of the function h that can be thought of as a privacy amplifier and secondly, the last set of measurement is performed in the Hadamard basis (as opposed to $|\pm_\alpha\rangle$ basis, for random α chosen by client in the $\mathsf{HBC-QFactory}$ case).

Then, it can be proven that, in an honest run, this output state is:

$$
\begin{aligned}
&|\mathtt{out}\rangle = H^{B_1} X^{B_2} |0\rangle \text{ , where} \\
&B_1 = h(x) \oplus h(x') = d_0(t_k) =: d_0 \\
&B_2 = (d_0 \cdot \langle b\,,\, (x \oplus x')\rangle) \oplus h(x)h(x')
\end{aligned}
$$

where $x \oplus x'$ denotes the bitwise xor of x and x' and $\langle \alpha\,,\, \beta\rangle$ denotes the inner product between the strings α and β: $\langle \alpha\,,\, \beta\rangle = \sum_i \alpha_i \cdot \beta_i$.

Therefore, the client can efficiently obtain the description of the output state, namely B_1 and B_2 by inverting y, to obtain the 2 preimages x and x' using her secret trapdoor information t_k.

Security. Informally speaking the desired security property of the module is to prove that the server cannot guess better than randomly the basis bit B_1 of what the client has, no matter how the server deviates or what answers he returns. In other words, we prove that given that the client chooses k randomly, then no matter which messages y and b the server returns, he cannot determine B_1.

Specifically, using the properties of the 2 cryptographic functions, we show that the basis of the output state is independent of the messages sent by server and essentially, the basis is fixed by the client at the beginning of the protocol.

Here it is important to emphasize that the simplicity of our modular construction allow us to make a direct reduction from the above security property to the cryptographic assumptions of our primitives functions $\mathcal{F}$, d_0 and h. Indeed, from the expression above, we can see that at the end of the interaction the client has recorded as the basis bit the expression $B_1 = h(x) \oplus h(x') = d_0(t_k)$, which is a hardcore bit and is therefore hard to guess given only k.

The Primitive Construction. In order to use this module in practise, it is crucial to have functions that satisfy our cryptographic requirements, and explore the choices of parameters that ensure that all these properties are jointly satisfied. Building on the function construction from chapter 4 we gave specific choices that achieve these properties. The starting point is the injective, trapdoor one-way family of functions $\bar{\mathcal{G}}$ from [MP12], where the hardness of the function is derived from the Learning-With-Errors problem.

More precisely, to sample a function f_k, we first sample a matrix $K \in \mathbb{Z}_q^{m \times n}$ using the construction of [MP12] (that provides an injective and trapdoor function), a uni-

form vector $s_0 \in \mathbb{Z}_q^n$, an error $e_0 \in \mathbb{Z}_q^m$ according to a small Gaussian[1] and a random bit d_0, and we compute

$$y_0 = Ks_0 + e_0 + d_0 \cdot \left(\frac{q}{2} \quad 0 \quad \dots \quad 0\right)^T \bmod q \tag{5.1}$$

The hardcore property of d_0 will directly come from the fact that under LWE assumption, no adversary can distinguish a LWE instance $Ks_0 + e_0$ from a random vector, so it is not possible to know if we added or not a constant vector. The function f_{K,y_0} will then be defined as follow:

$$f_{K,y_0}(s,e,c,d) = Ks + e + c \cdot y_0 + d \cdot \left(\frac{q}{2} \quad 0 \quad \dots \quad 0\right)^T \bmod q \tag{5.2}$$

Note that c and d are bits, and the error e is chosen in a bigger space[2] than e_0 to ensure that the function f_{K,y_0} has two preimages with good probability. Moreover, if we define $h(s,e,c,d) = d$, it is easy to see that for all preimages x, x' with $f(x) = f(x')$, we have:

$$h(x) \oplus h(x') = d_0$$

The Extended Protocol. In order to use the above protocol for applications such as blind quantum computing [BFK09], we need to be able to produce states taken from the (extended) set of eight states $\{|+_\theta\rangle, \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\}$. Importantly, we still need to ensure that the bits corresponding to the basis of each qubits produced, remain hidden. Here we prove how given two states produced by the basic protocol described previously, which we denote as $|\mathtt{in}_1\rangle$ and $|\mathtt{in}_2\rangle$, we can obtain a single state from the 8-states set, denoted $|out\rangle$, ensuring that no information about the bits of the basis of $|out\rangle$ is leaked[3].

To achieve this, we need to find an operation (see Figure 5.2 in Section 5.4.1), that in the honest case maps the indices of the inputs to those of the output using a map that satisfies certain conditions. This relation (inputs/output) should be such that learning anything about the basis of the output state implies learning non-negligible information for the basis of (one) input. This directly means, that any computationally bounded adversary that can break the basis blindness of the output, can use this to construct an attack that would also break the basis blindness of at least one of the inputs, i.e. he would break the security guarantees of the basic module that was proven earlier.

[1] but big enough to make sure the function is secure

[2] but small enough to make sure the partial functions $f(\cdot,\cdot,c,\cdot)$ are still injective

[3] Note that one of the input states is exactly the output of the basic module, while the second comes from a slightly modified version (essentially rotated in the XY-plane of the Bloch sphere).

Other Properties. To further improve the practicality of the black box call of the QFactory we also present the security against abort scenario that could be achieved based on a quantum version of Yao's XOR Lemma.

Before, describing the main protocol, we first need to introduce some notation used in the remaining of the chapter.

5.1.1 Notations

For a state $|+_\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$, where $\theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}$, we use the notation:

$$\theta = \frac{\pi}{4} \cdot L$$

Additionally, as L is a 3-bit string, we write it as $L = L_1L_2L_3$, where L_1, L_2, L_3 represent the bits of L.
As a result when we refer to the basis of the $|+_\theta\rangle$ state, it is equivalent to referring to the last 2 bits of L, thus saying that nothing is leaked about the basis of this state, is equivalent to saying nothing is leaked about the bits L_2 and L_3.
For a set of 4 quantum states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$, we denote the index of each state using 2 bits: B_1, B_2, with $B_1 = 0$ if and only if the state is $|0\rangle$ or $|1\rangle$, and $B_2 = 0$ if and only if the state is $|0\rangle$ or $|+\rangle$, i.e. $H^{B_1}X^{B_2}|0\rangle$. We will use interchangeably the Dirac notation and the basis/value notation.

In the following sections, we will consider polynomially bounded malicious adversaries, usually denoted by $\mathcal{A}$. The honest clients will be denoted by π, and both honest parties and adversaries can output some values, that could eventually be used in other protocols. To denote that two parties π_A and $\mathcal{A}$ interact in a protocol, and that π_A outputs a while $\mathcal{A}$ outputs b, we write $(a, b) \leftarrow (\pi_A \| \pi_B)$ (we may forget the left hand side, or replace variables with underscores "_" if it is not relevant). We can also refer to the values of the classical messages sent between the two parties using: $\Pr[a = \texttt{accept} \mid (\pi_A \| \mathcal{A})]$, and this probability is implicitly over the internal randomness of π_A and $\mathcal{A}$. To specify a two-party protocol, it suffices to specify the two honest parties (π_A, π_B). Moreover, if the protocol consists of a single round of communication, we will write $y \leftarrow \mathcal{A}(x)$ with x the first message sent to $\mathcal{A}$, and y the messages sent from $\mathcal{A}$. Finally, a value with a tilde, such as $\tilde{d}$, represents a guess from an adversary.

We are considering protocols secure against quantum adversaries, so we assume that all the properties of our functions hold against a general Quantum Polynomial

Time (QPT) adversary. We will denote $\mathcal{D}$ the domain of the functions, while $\mathcal{D}(n)$ is the subset of strings of length n.

5.2 The Malicious 4-states QFactory Protocol

5.2.1 Requirements and protocol

The Malicious 4-states QFactory Protocol described in Protocol 3 uses a family of cryptographic functions $\mathcal{F}$ and a function h having the following properties (see section 5.3 to see how this family of functions can be constructed from a family of injective, trapdoor and (pseudo) homomorphic functions):

Definition 5.2.1 (2-regular homomorphic-hardcore family)**.** *A family $\mathcal{F} = \{f_k : \mathcal{D}' \to \mathcal{R}\}_{k\in\mathcal{K}}$ is said to be a 2-regular homomorphic-hardcore family with respect to $h_k : \mathcal{D}' \to \{0,1\}$ and $d_0 : \mathcal{T} \to \{0,1\}$ ($\mathcal{T}$ is the set of trapdoors t_k) if:*

- *it is 2-regular, collision resistant and trapdoor;*
- *for all k, h_k can be described by a polynomial classical circuit;*
- *d_0 is a hardcore predicate for* Pub*, i.e. given a random index $k = \mathtt{Pub}_{\mathcal{F}}(t_k)$, it is impossible to obtain $d_0 := d_0(t_k)$ with probability better than $1/2 + \mathsf{negl}(n)$, i.e. for any QPT adversary $\mathcal{A}$:*

$$\Pr[\mathcal{A}(k) = d_0(t_k) \mid (k,t_k) \leftarrow \mathtt{Gen}_{\mathcal{F}}] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{5.3}$$

- *for all $k \in \mathcal{K}$ and $x, x' \in \mathcal{D}'$ such that $f_k(x) = f_k(x')$, we have:*

$$h_k(x) \oplus h_k(x') = d_0 \tag{5.4}$$

Note that in our specific construction h does not depend on k, so we might omit the subscript k, and just use h, for simplicity.
We also extend this definition to δ-2-regular homomorphic-hardcore family, when the function is δ-2-regular, i.e. 2-regular with probability δ (see Definition B.4.2 for a formal definition).

Then the protocol can be described as follows:

Protocol 3 Malicious 4-states QFactory Protocol: classical delegation of BB84 states

Requirements:

Public: A δ-2-regular homomorphic-hardcore family $\mathcal{F}$ with respect to $\{h_k\}$ and d_0, as described above. For simplicity, we will represent the sets $\mathcal{D}'$ (respectively $\mathcal{R}$) using n (respectively m) bits strings: $\mathcal{D}' = \{0,1\}^n$, $\mathcal{R} = \{0,1\}^m$. In this protocol, we require δ to be negligibly

close to 1, see section 5.5 for an extension to a constant δ.

Stage 1: Preimages superposition

– Client: runs the algorithm $(k, t_k) \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^n)$.

– Client: instructs Server to prepare one register at $\otimes^n H|0\rangle$ and a register initiated at $|0\rangle^m$.

– Client: sends k to Server and the Server applies U_{f_k} using the first register as control and the second as target.

– Server: measures the second register in the computational basis, obtains the outcome y. Here, in an honest run, the Server would have a state $(|x\rangle + |x'\rangle) \otimes |y\rangle$ with $f_k(x) = f_k(x') = y$ and $y \in \mathrm{Im}\, f_k$.

Stage 2: Output preparation

– Server: applies U_{h_k} on the preimage register $|x\rangle + |x'\rangle$ as control and another qubit initiated at $|0\rangle$ as target. Then, measures all the qubits, but the target in the $\{\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)\}$ basis, obtaining the outcome $b = (b_1, ..., b_n)$. Now, the Server returns both y and b to the Client.

– Client: using the trapdoor t_k computes the preimages of y:

- if y does not have exactly two preimages x, x' (the server is cheating with overwhelming probability), defines $B_1 = d_0(t_k)$, and chooses $B_2 \in \{0,1\}$ uniformly at random
- if y has exactly two preimages x, x', defines $B_1 = h_k(x) \oplus h_k(x') = d_0(t_k)$, and B_2 as defined in Theorem 5.2.2.

Output: If the protocol is run honestly, the state that the Server has produced is (with overwhelming probability) the BB84 state $|\mathtt{out}\rangle = H^{B_1} X^{B_2} |0\rangle$, having the basis $B_1 = h_k(x) \oplus h_k(x') = d_0$ (see Theorem 5.2.2 for the exact value of B_2). The output of the Server is $|\mathtt{out}\rangle$, and the output of the Client is (B_1, B_2).

5.2.2 Correctness of Malicious 4-states QFactory

In an honest run, the description of the output state of the protocol depends on measurement results $y \in \mathrm{Im}\, f_k$ and b, but also on the 2 preimages x and x' of y.

The output state of Malicious 4-states QFactory belongs to the set of states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ and its exact description is the following:

Theorem 5.2.2. *In an honest run, with overwhelming probability the output state $|\mathtt{out}\rangle$ of the Malicious 4-states QFactory Protocol (Protocol 3) is a BB84 state whose basis is $B_1 = h_k(x) \oplus h_k(x') = d_0$, and:*

- *if $d_0 = 0$, then the state is $|h_k(x)\rangle$ (computational basis, also equal to $|h_k(x')\rangle$)*
- *if $d_0 = 1$, then if $\sum_i b_i \cdot (x_i \oplus x'_i) = 0 \bmod 2$, the state is $|+\rangle$, otherwise the state is $|-\rangle$ (Hadamard basis).*

i.e.

$$|\mathtt{out}\rangle = H^{B_1} X^{B_2} |0\rangle \tag{5.5}$$

with

$$B_1 = h_k(x) \oplus h_k(x') = d_0 \tag{5.6}$$

$$B_2 = (d_0 \cdot \langle b\,,\, (x \oplus x')\rangle) \oplus h(x)h(x') \tag{5.7}$$

(the inner product is taken modulo 2, and $x \oplus x'$ is a bitwise xor)

Proof. The operations performed by the quantum server, can be observed in Fig. 5.1:

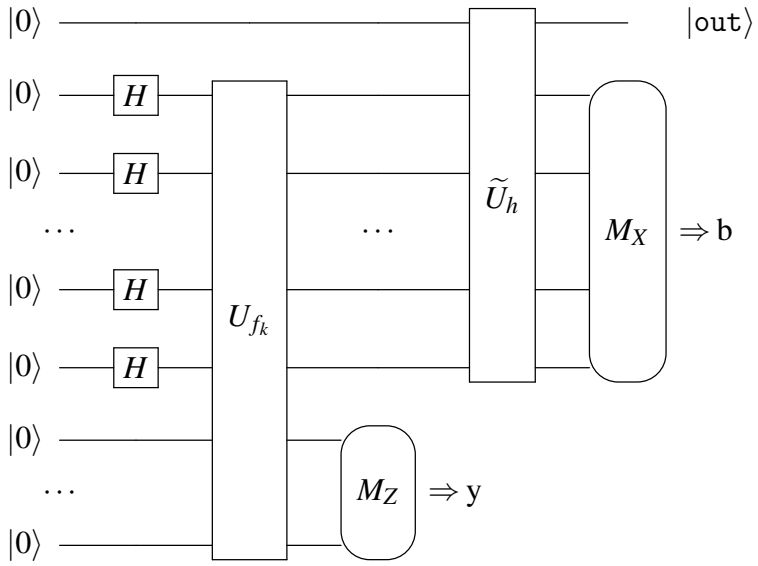


Figure 5.1: The circuit computed by the Server

$$|0\rangle \otimes |0^n\rangle \otimes |0^m\rangle \xrightarrow{I_2 \otimes H^{\otimes n} \otimes I_2^{\otimes m}} |0\rangle \otimes \sum_{x \in \mathcal{D}} |x\rangle \otimes |0^m\rangle \xrightarrow{I_2 \otimes U_{f_k}}$$

$$|0\rangle \otimes \sum_{x \in \mathcal{D}} |x\rangle \otimes |f_k(x)\rangle \xrightarrow{f_k \text{2-regular}} |0\rangle \otimes \sum_{y \in \mathrm{Im}(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \xrightarrow{I_2 \otimes I_2^{\otimes n} \otimes M_Z^{\otimes m}}$$

$$|0\rangle \otimes (|x\rangle + |x'\rangle) \otimes |y\rangle \xrightarrow{\widetilde{U}_h \otimes I_2^{\otimes m}} (|h(x)\rangle \otimes |x\rangle + |h(x')\rangle \otimes |x'\rangle) \otimes |y\rangle \xrightarrow{I_2 \otimes M_X^{\otimes n} \otimes I_2^{\otimes m}}$$

$$|\mathtt{out}\rangle \otimes |b_1\rangle \ldots \otimes |b_n\rangle \otimes |y\rangle \Rightarrow |\mathtt{out}\rangle = H^{d_0} X^{(d_0 \cdot \langle b \cdot (x \oplus x')\rangle) \oplus h(x)h(x')} |0\rangle$$

where $\widetilde{U}_h$ is a "swapped" U_h, acting on the first register as target and input register as control: $|0\rangle |x\rangle \xrightarrow{\widetilde{U}_h} |h(x)\rangle |x\rangle$.

The server initially prepares the state $|0^n\rangle \otimes |0^m\rangle$, where we will call the first register the preimage register, and the second one the image register.
After applying U_{f_k} we obtain the state $\sum_{x\in\mathcal{D}} |x\rangle |f_k(x)\rangle$. Using the 2-regularity property of f_k, after measuring the second register (in the computational basis) and obtaining the measurement result $y \in \mathrm{Im}(f_k)$, the state can be expressed as $(|x\rangle + |x'\rangle) \otimes |y\rangle$, where x and x' are the 2 unique preimages of y. By omitting the image register and by initializing another qubit in the $|0\rangle$ state and using the above notation, the input to the unitary $\widetilde{U}_h$ can be written as:

$$(|x\rangle + |x'\rangle) \otimes |0\rangle \tag{5.8}$$

$\widetilde{U}_h$ is basically U_h acting on the input and the new register, and after we apply it, we obtain the state:

$$(|x\rangle \otimes |h(x)\rangle + |x'\rangle \otimes |h(x')\rangle \tag{5.9}$$

As a final step, we measure all but the last qubit of this state in the $\{\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)\}$ basis (obtaining the measurement result string b), which is equivalent to applying $H^{\otimes n}$ on the input register, and then measuring it in the computational basis. Thus, after applying the Hadamard gates (which is a Fourier transformation in $\mathbb{Z}_2$), we get:

$$\sum_{b=1}^{2^n-1} (-1)^{\langle b,x\rangle} |b\rangle \otimes |h(x)\rangle + \sum_{b=1}^{2^n-1} (-1)^{\langle b,x'\rangle} |b\rangle \otimes |h(x')\rangle$$

After obtaining the measurement result b, the remaining state becomes (up to a global phase):

$$\begin{aligned} |\mathtt{out}\rangle &= (-1)^{\langle b,x\rangle} |h(x)\rangle + (-1)^{\langle b,x'\rangle} |h(x')\rangle \\ &= |h(x)\rangle + (-1)^{\langle b,(x\oplus x')\rangle} |h(x')\rangle \end{aligned} \tag{5.10}$$

Therefore, we have:

- if $h(x) = h(x')$ (i.e. $d_0 = h(x) \oplus h(x') = 0$, using Equation 5.4), we have $|\mathtt{out}\rangle = |h(x)\rangle = H^{d_0} X^{h(x)} |0\rangle$
- if $h(x) \neq h(x')$ (i.e. $d_0 = h(x) \oplus h(x') = 1$) we have $|\mathtt{out}\rangle = |+\rangle$ iff $\langle b, (x \oplus x')\rangle = 0 \bmod 2$, and $|-\rangle$ otherwise. Thus, $|\mathtt{out}\rangle = H^{d_0} X^{\langle b,(x\oplus x')\rangle} |0\rangle$

Hence, $|\mathtt{out}\rangle = H^{B_1} X^{B_2}$ with $B_1 = d_0 = h(x) \oplus h(x')$, and

$$\begin{aligned} B_2 &= (1 \oplus h(x) \oplus h(x'))h(x) + (h(x) \oplus h(x')) \cdot \langle b, (x \oplus x')\rangle \bmod 2 \\ &= h(x) + h(x)^2 + h(x)h(x') + d_0 \cdot \langle b, (x \oplus x')\rangle \bmod 2 \\ &= h(x) + h(x) + h(x)h(x') + d_0 \cdot \langle b, (x \oplus x')\rangle \bmod 2 \\ &= h(x)h(x') \oplus d_0 \cdot \langle b, (x \oplus x')\rangle \bmod 2 \end{aligned}$$

□

It can be noticed that, in an honest run of the protocol, using y and the trapdoor information of the function f_k, the Client obtains x and x' and thus can efficiently determine what is the output state that the Server has prepared.
In the next section, we prove that no malicious adversary can distinguish between the 2 possible bases $\{|0\rangle, |1\rangle\}$ and $\{|+\rangle, |-\rangle\}$ of the output qubit, or equivalently distinguish whether B_1 is 0 or 1.

5.2.3 Security of Malicious 4-states QFactory

In any run of the protocol, honest or malicious, the state that the client believes that the server has is given by Theorem 5.2.2. Therefore, the task that a malicious server wants to achieve, is to be able to guess, as good as he can, the description of the output state that the client (based on the public communication) thinks the server has produced. In particular, in our case, the server needs to guess the bit B_1 (corresponding to the basis) of the (honest) output state.

Note that we want to make sure that the server cannot guess the basis bit B_1 (for most applications ([BFK09, FK17]) basis blindness is sufficient as indicated in [DK16]), and we do not care about the value bit B_2 simply because it is not possible to say that B_2 cannot be guessed with probability better than random. Indeed, even in the honest case, or in the "perfect" case with a quantum channel, the server can always measure the qubit $|\mathtt{out}\rangle$ he has to extract the value bit (for example by measuring it in a random basis (computational or Hadamard) and outputting the outcome of the measurement, he will succeed with probability $\frac{1}{2}\cdot\frac{1}{2}+\frac{1}{2}\cdot 1=\frac{3}{4}>1/2$). Additionally, partial blindness of B_2 is implicit in our work, since learning B_2 leads to leaking partial information about B_1, in the case that the server possesses the honest output state $H^{B_1}X^{B_2}|0\rangle$. Optimal bounds for B_2's leakage are not known if the server is malicious and without verification, is non-trivial and will be studied as a future work.

Definition 5.2.3 (4 states basis blindness)**.** *We say that a protocol (π_A,π_B) achieves **basis-blindness** with respect to an ideal list of 4 states $S=\{S_{B_1,B_2}\}_{(B_1,B_2)\in\{0,1\}^2}$ if:*

- *S is the set of states that the protocol outputs, i.e.:*

$$\Pr\left[|\phi\rangle = S_{B_1B_2}\in S \mid ((B_1,B_2),|\phi\rangle)\leftarrow(\pi_A\|\pi_B)\right]\geq 1-\mathsf{negl}(n)$$

- *No information is leaked about the index bit B_1 of the output state of the protocol, i.e for all QPT adversary $\mathcal{A}$:*

$$\Pr\left[B_1 = \tilde{B_1} \mid ((B_1,B_2),\tilde{B_1}) \leftarrow (\pi_A \| \mathcal{A})\right] \leq 1/2 + \mathsf{negl}(n)$$

Theorem 5.2.4 (Malicious 4-states QFactory is secure)**.** *Protocol 3 satisfies 4-states basis blindness with respect to the ideal list of states* $S = \{H^{B_1}X^{B_2}|0\rangle\}_{B_1,B_2} = \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$.

Proof. The advantage of our construction is that this theorem is now a direct application of the definition of the family $\mathcal{F}$ (Definition 5.2.1). Indeed, let us suppose that there exists a QPT adversary $\mathcal{A}$ such that:

$$\Pr\left[B_1 = \tilde{B_1} \mid ((B_1,B_2),\tilde{B_1}) \leftarrow (\pi_A \| \mathcal{A})\right] \geq 1/2 + \frac{1}{\mathsf{poly}(n)}$$

where π_A is the honest Client and $\mathcal{A}$ is the malicious Server in Protocol 3.
From Theorem 5.2.2, we notice that the value of B_1 is always equal to $d_0(t_k)$. Moreover, our adversary is a one-round adversary, so we can rewrite the previous equation as:

$$\Pr[d_0(t_k) = \mathcal{A}(k) \mid (k,t_k) \leftarrow \mathtt{Gen}_{\mathcal{F}}] \geq 1/2 + \frac{1}{\mathsf{poly}(n)}$$

But d_0 is a hardcore predicate, so this contradicts Equation 5.3. So no QPT adversary $\mathcal{A}$ can guess the basis B_1 with probability better than $1/2 + \mathsf{negl}(n)$.

□

Remark 5.2.5. *In the run of the Malicious 4-states QFactory protocol, the adversary (server) has no access to the abort/accept bit, specifying whether the Client wants to abort the protocol after receiving the image y from the server (the abort occurs when y does not have exactly two preimages). So this is why this first protocol is correct with overwhelming probability only when the function f_k is 2-regular with overwhelming probability ($\delta > 1 - \mathsf{negl}(n)$). See section 5.5 to see how we address this issue for constant δ.*

5.3 Function Implementation

5.3.1 Generic construction of 2-regular homomorphic-hardcore

To complete the construction of Malicious 4-states QFactory, we must find functions $\mathcal{F}$, h, and d_0 satisfying the properties described in Definition 5.2.1. We first explain

a general method to construct a 2-regular function from an injective homomorphic function (the generalisation to δ-2-regularity from pseudo-homomorphic functions is treated in section B.4), and we give in the next section a candidate that achieves the two other properties required in our definition (homomorphic-hardcore predicate) whose security is based on the cryptographic problem LWE.

Lemma 5.3.1. *It is possible to construct a family of functions $\mathcal{F} : \{f_k : \mathcal{D} \times \{0,1\} \to \mathcal{R}\}$, h_k and d_0 that are a 2-regular homomorphic-hardcore family (Definition 5.2.1) from a family of functions $\mathcal{G} = \{g_{k'} : \mathcal{D} \to \mathcal{R}\}_{k'}$ that is:*

- *injective*
- *trapdoor*
- *homomorphic* [4]

and such that for every $g_{k'}$ there exists a homomorphic hardcore predicate $h'_{k'}$.[5]

Proof. Because $\mathcal{G}$ is homomorphic, there exist 2 operations "$+_{\mathcal{D}}$" acting on $\mathcal{D}$ and "$+_{\mathcal{R}}$" acting on $\mathcal{R}$ such that:

$$g_{k'}(z_1 +_{\mathcal{D}} z_2) = g_{k'}(z_1) +_{\mathcal{R}} g_{k'}(z_2) \; \forall k' \; \forall z_1, z_2 \in \mathcal{D} \tag{5.11}$$

The hardcore predicate $h' : \mathcal{D} \to \{0,1\}$ is homomorphic, hence we have:

$$h'(z_1) \oplus h'(z_2) = h'(z_2 +_{\mathcal{D}} z_1) \; \forall z_1, z_2 \in \mathcal{D} \tag{5.12}$$

And because we are working modulo 2 it is easy to see that:

$$h'(z_1) \oplus h'(z_2) = h'(z_2 -_{\mathcal{D}} z_1) \; \forall z_1, z_2 \in \mathcal{D}, \tag{5.13}$$

where "$-_{\mathcal{D}}$" is the inverse of the operation "$+_{\mathcal{D}}$".

Then, the functions $\mathcal{F}$, h_k, d_0 are constructed as follow. First, to generate a private key, we generate a private key of $\mathcal{G}$, and we pick a random element z_0 from $\mathcal{D}$:

[4] We only require $\mathcal{G}$ to be homomorphic with good probability for a single application of the operation $+_{\mathcal{D}}$ and this would result in $\mathcal{F}$ being 2-regular with good probability, as proven in section B.4.

[5] We will omit the subscript k' and simply denote the hardcore predicate by h' - as we will see in the next section the instantiation of h is independent of k'.

$\mathtt{Gen}_{\mathcal{F}}(1^n)$

1 : $(k', t'_{k'}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$

2 : $z_0 \leftarrow_{\$} \mathcal{D}$

3 : $y_0 \leftarrow g_{k'}(z_0)$

4 : $t_k = (t'_{k'}, z_0)$

5 : $k = (k', y_0)$

6 : **return** (k, t_k)

And then we define $f_k : \mathcal{D} \times \{0,1\} \to \mathcal{R}$ as:

$$f_k(z,c) = g_{k'}(z) +_{\mathcal{R}} c \cdot y_0$$

We also define $h_k : \mathcal{D} \times \{0,1\} \to \mathcal{R}$ as:

$$h_k(z,c) = h(z)$$

and $d_0 : \mathcal{T} \to \{0,1\}$ (where $\mathcal{T}$ is the sets of trapdoors t_k) as

$$d_0(t_k) = h(z_0)$$

Now we need to check the properties of $\mathcal{F}$, h_k and d_0 from Definition 5.2.1:

- The 2-regularity of $\mathcal{F}$ comes directly from the injectivity of $\mathcal{G}$: for any $y \in \mathrm{Im}\, f_k$, we have one preimage $(g_{k'}^{-1}(y), 0)$ and one preimage $(g_{k'}^{-1}(y) - z_0, 1)$. The past two formula also show the trapdoor property using the fact that $\mathcal{G}$ is a trapdoor family (more details can be found in Theorem 4.5.1).

- The collision-resistant property comes from the homomorphicity , injectivity, and one-wayness of $\mathcal{G}$: if we find a collision, we can write a reduction that breaks the one-wayness of $\mathcal{G}$. Indeed by injectivity two different preimages have a different c, i.e. $f_k(z,0) = f_k(z',1)$. Hence, $g_{k'}(z) = g_{k'}(z') + g_{k'}(z_0)$, i.e. $z_0 = z - z'$. So it means that from a collision we can find z_0, which contradicts the one-wayness of $g_{k'}$.

- h_k is equal to h, so it can be computed efficiently.

- $d_0(t_k) = h(z_0)$, and h is a hardcore predicate, so d_0 is also hardcore predicate

- The last condition is respected, because if $f_k(z,0) = f_k(z',1)$, we have:

$$\begin{aligned} h_k(z,0) \oplus h_k(z',1) &= h(z) \oplus h(z') \\ &= h(z -_{\mathcal{R}} z') \\ &= h(z_0) \\ &= d_0(t_k) = d_0 \end{aligned}$$

□

5.3.2 Construction of δ-2-regular homomorphic-hardcore family $\mathcal{F}$

We will now give an explicit implementation of a family $\mathcal{G}$ that is injective, trapdoor, (pseudo) homomorphic with a homomorphic-hardcore predicate d_0, and then we will rely on a construction similar to Lemma 5.3.1 to produce a family $\mathcal{F}$, h, and d_0 with the properties described in Definition 5.2.1 needed by Protocol 3 and Protocol 5. We notice that we defined in the previous chapter a similar construction, but without the additional homomorphic-hardcore property.

The starting point is the injective, trapdoor one-way family of functions $\bar{\mathcal{G}} = \{\bar{g}_K : \mathbb{Z}_q^n \times E^m \to \mathbb{Z}_q^m\}_K$[6] from [MP12] (where E defines the set of integers bounded in absolute value by some "big-enough" value μ which will be defined later, and additions are matrix additions modulo q, where q is an even integer).

$$\bar{g}_K(s,e) = Ks + e$$

Then, to sample a function from the family $\mathcal{G} = \{g_{k'} : \mathbb{Z}_q^n \times E^m \times \{0,1\} \to \mathbb{Z}_q^m\}$, which will be used for construction $\mathcal{F}$ as in Lemma 5.3.1, we first build the public key matrix $K \in \mathbb{Z}_q^{m \times n}$ along with the trapdoor matrix R using the construction from [MP12] and additionally, we extend the domain with an extra input bit d. Now, after denoting the constant vector $v = \begin{pmatrix} \frac{q}{2} & 0 & \dots & 0 \end{pmatrix}^T$, we define $g_{k'}$ as:

$$g_{k'}(s,e,d) = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = Ks + e + d \cdot v \tag{5.14}$$

where $k' := K$ and $t'_{k'} := R$

[6] The bar on top of $\bar{\mathcal{G}}$ denotes the version where there is not yet the hardcore bit d_0

Finally, to sample a function from the family $\mathcal{F} = \{f_k : \mathbb{Z}_q^n \times E^m \times \{0,1\} \times \{0,1\} \to \mathbb{Z}_q^m\}$, we construct matrices K and R as before, and will sample a uniform random vector $s_0 \in \mathbb{Z}_q^n$, a random small error vector $e_0 \in \mathbb{Z}_q^m$ sampled according to a "small-enough" Gaussian distribution $\mathcal{D}_{\alpha' q}^m$ on integers and a (uniform) random bit $d_0 \in \{0,1\}$. Now by defining:

$$y_0 := Ks_0 + e_0 + d_0 \cdot v$$

the trapdoor is set to $t_k := (R, s_0, e_0, d_0)$, and the public index is $k := (K, y_0)$.

We can already note at this step that d_0 is a hardcore-predicate:

Lemma 5.3.2. *The function $d_0(t_k) := d_0$ is a hardcore predicate of k, i.e. for all QPT adversaries $\mathcal{A}$, we have:*

$$\Pr[\mathcal{A}(k) = d_0(t_k)] \leq \frac{1}{2} + \mathsf{negl}(n)$$

The proof can be found in subsection B.1.1.
Now, we can define $f_k : \mathbb{Z}_q^n \times E^m \times \{0,1\} \times \{0,1\} \to \mathbb{Z}_q^m$ as follows:

$$f_k(s,e,c,d) = Ks + e + c \cdot y_0 + d \cdot v$$

and $h : \mathbb{Z}_q^n \times E^m \times \{0,1\} \times \{0,1\} \to \{0,1\}$ as:

$$h(s,e,c,d) = d$$

The idea behind this construction is similar to the general construction presented in subsection 5.3.1. Intuitively, the first two terms $Ks+e$ are useful for the security, the $c \cdot y_0$ term is needed to ensure the 2-regularity (the two images will differ by $(s_0, e_0, 1, d_0)$), and the last term $d \cdot v$ is mostly useful to provide the hardcore property. More precisely:

- This function cannot have more than 2 preimages because the partial functions $f(\cdot,\cdot,c,\cdot)$ are injective (because $\bar{g}_K$ is injective).
- h is the homomorphic-hardcore predicate required by Definition 5.2.1. Indeed, if there is a collision, i.e. if $f_k(s,e,0,d) = f_k(s',e',1,d')$, it is easy to see that $d \oplus d' = d_0$ (q is even, and operations are modulo q), i.e.: $h(x) \oplus h(x') = d_0$.
- Finally, for an appropriate choice of parameters (see Lemma 5.3.3), this function is 2-regular with good probability. Indeed, if for a random element $(s,e,0,d)$ there exists $(s',e',1,d')$ with $f_k(s,e,0,d) = f_k(s',e',1,d')$, then $e = e' + e_0$. But e_0 is sampled from a set significantly smaller than E, so with good probability $e' = e - e_0$ will belong to E.

Note on the parameters: α' is chosen to make sure that the sampled elements are small compared to μ (the upper bound on E), but such that the noise is still big enough for security. On the contrary, μ must stay small enough to ensure that the function does not have more than two preimages. In the previous chapter we provided a set of parameters having all the required constraints, which we recall here:

Lemma 5.3.3. *The family of functions $\mathcal{F}$ is δ-2-regular with good (constant greater than 1/2) probability, trapdoor, one-way and collision resistant (all these properties are true even against a quantum attacker), assuming that there is no quantum algorithm that can efficiently solve* SIVP_γ *for $\gamma = \mathsf{poly}(n)$, for the following choices of parameters:*

$$\begin{aligned} q &= 2^{5\lceil log(n)\rceil+21} \\ m &= 23n + 5n\lceil log(n)\rceil \\ \mu &= 2mn\sqrt{23+5log(n)} \\ \alpha' &= \frac{\mu}{m\sqrt{m}q} \end{aligned} \tag{5.15}$$

Moreover, we can find another set of parameters such that this probability δ is negligibly close to one assuming that SIVP_γ *is secure for a superpolynomial γ (depending on the value of δ, you may choose Protocol 3 ($\delta \sim 1$) or Protocol 5 ($\delta > 1/2$)).*

We can now formalize the above intuitions:

Theorem 5.3.4. *The family $\mathcal{F}$ defined above with appropriate parameters, such as the one defined in Lemma 5.3.3, is a δ-2-regular homomorphic-hardcore family.*

Proof. Le us first show that g_K is an injective function. To prove this, we will use the injectivity property of the function $\bar{g}$.

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v$$

Assume there exist 2 tuples (s_1,e_1,d_1) and (s_2,e_2,d_2) such that $g_K(s_1,e_1,d_1) = g_K(s_2,e_2,d_2)$. To prove that g_K is injective we must show that $(s_1,e_1,d_1) = (s_2,e_2,d_2)$. This is equivalent to:

$$\begin{aligned} \bar{g}_K(s_1,e_1) + d_1 \cdot v &= \bar{g}_K(s_2,e_2) + d_2 \cdot v \\ \bar{g}_K(s_1,e_1) - \bar{g}_K(s_2,e_2) + (d_1 - d_2) \cdot v &= 0 \end{aligned} \tag{5.16}$$

Now, if $d_1 = d_2$, then we have that $\bar{g}_K(s_1,e_1) - \bar{g}_K(s_2,e_2) = 0$, and because $\bar{g}_K$ is injective, this would imply that $s_1 = s_2$ and $e_1 = e_2$ and thus, g_K would also be injective.

Let us suppose that $d_1 \neq d_2$ and we will prove that this is impossible. Without loss of generality we can assume that $d_1 = 0$ and $d_2 = 1$.
Thus, we want to show that it is impossible to have (s_1, e_1) and (s_2, e_2) such that:

$$\bar{g}_K(s_1, e_1) - \bar{g}_K(s_2, e_2) = v \tag{5.17}$$

Using the definition of $\bar{g}_K$, this is equivalent to:

$$Ks_1 + e_1 - Ks_2 - e_2 = \begin{pmatrix} \frac{q}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \bmod q \tag{5.18}$$

This can be rewritten as:

$$K_{1,1}(s_{1,1} - s_{2,1}) + ... + K_{1,n}(s_{1,n} - s_{2,n}) + (e_{1,1} - e_{2,1}) = \frac{q}{2} \bmod q \tag{5.19}$$

$$K_{i,1}(s_{1,1} - s_{2,1}) + ... + K_{i,n}(s_{1,n} - s_{2,n}) + (e_{1,i} - e_{2,i}) = 0 \bmod q \tag{5.20}$$

for any $i \in \{2, \cdots, m\}$ and where $s_{1,i}$ and $s_{2,i}$ denote the i-th elements of s_1 and s_2 respectively and $e_{1,i}$ and $e_{2,i}$ are the i-th elements of e_1, respectively e_2.
Now, as the function $\bar{g}_K : \mathbb{Z}_q^n \times E^m \to \mathbb{Z}_q^m$, where $K \leftarrow \mathbb{Z}_q^{m \times n}$, is injective, the following function is also injective:
$\bar{g_1}_{K_1} : \mathbb{Z}_q^n \times E^{m-1} \to \mathbb{Z}_q^{m-1}$, where $K_1 \leftarrow \mathbb{Z}_q^{m-1 \times n}$ and where $\bar{g}$ and $\bar{g_1}$ have the exact same definition:

$$\bar{g_1}_{K_1}(s, e) = K_1 s + e \bmod q \tag{5.21}$$

More specifically, the only difference between the 2 functions is the change of dimension from m to $m-1$, but as the injectivity proof from [MP12] holds for any $m = \Omega(n)$, then $\bar{g_1}$ is also injective.
Now, consider the matrix K_1 obtained from K by removing the first line. As shown above, $\bar{g_1}_{K_1}$ is an injective function, thus from Equation 5.20, we get that:

$$s_1 = s_2 \tag{5.22}$$

$$e_{1,i} = e_{2,i} \, \forall i = 2, .., m \tag{5.23}$$

Now as $s_1 = s_2$, from Equation 5.19, we obtain $e_{1,1} - e_{2,1} = \frac{q}{2}$.
However, from the domain of $\bar{g}$, we have that: $|e_{1,1} - e_{2,1}| < |e_{1,1}| + |e_{2,1}| < 2\mu < \frac{q}{2}$ (where for the last inequality we used Lemma 5.3.3). This concludes the contradiction proof and shows us that g_K is injective.

The proofs of homomorphicity for g_K and h can be found in subsection B.1.2 and one-wayness of g_K in subsection B.1.3. Finally, the Lemma 5.3.3 ensures that the family is δ-2-regular and the hardcore property results from Lemma 5.3.2.

□

5.4 The Malicious 8-states QFactory Protocol

In order to use the Malicious 4-states QFactory Protocol functionality for applications such as blind quantum computing [BFK09], we need to be able to produce states taken from the set $\{|+_\theta\rangle, \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\}$, always ensuring that the bases of these qubits remain hidden. Here we prove how by using two states produced by Malicious 4-states QFactory Protocol, we can obtain a single state from the 8-states set, while no information about the bases of the new output state is leaked.

To achieve this, we need to find an operation, that in the honest case maps the correct inputs to the outputs, in such a way, that the index of the output state corresponding to the basis, is directly related with the bases bits of the input states. This relation should be such that learning non-negligible information about the basis of the output state implies learning non-negligible information about the basis of the input state. This directly means, that any computationally bounded adversary that breaks the 8-states basis blindness of the output, also breaks the 4-states basis blindness of at least one of the inputs.

The protocol achieving this task is described as outlined below.

Protocol 4 Malicious 8-states QFactory

Requirements: Same as in Protocol Protocol 3

Input: Client runs 2 times the algorithm $Gen_{\mathcal{F}}(1^n)$, obtaining $(k^1, t_k^1), (k^2, t_k^2)$. Client keeps t_k^1, t_k^2 private.

Protocol:

– Client: runs Malicious 4-states QFactory algorithm to obtain a state $|\mathtt{in}_1\rangle$ and a "rotated" Malicious 4-states QFactory to obtain a state $|\mathtt{in}_2\rangle$. By a rotated Malicious 4-states QFactory we mean a Malicious 4-states QFactory outputting one of the 4 states $\{|+\rangle, |-\rangle, \left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$. This can be obtained from the construction of Malicious 4-states QFactory, by simply applying the operation $HR\left(-\frac{\pi}{2}\right)$ on the BB84 output state.

– Client: records measurement outcomes (y^1, b^1), (y^2, b^2) and computes and stores the corresponding indices of the output states of the 2 Malicious 4-states QFactory runs: (B_1, B_2) for $|\mathtt{in}_1\rangle$ and (B_1', B_2') for $|\mathtt{in}_2\rangle$.

– Client: instructs Server to apply Merge Gadget (Figure 5.2) on the states $|\mathtt{in}_1\rangle$, $|\mathtt{in}_2\rangle$.

– Server: returns the 2 measurement results s_1, s_2.

– Client: using (B_1,B_2), (B_1',B_2'), s_1, s_2 computes the index $L = L_1L_2L_3 \in \{0,1\}^3$ of the output state (as showed in Eq.(5.29), Eq.(5.30) and Eq.(5.31) of Theorem 5.4.1).

Output: If the protocol is run honestly, the state that the Server has produced is:

$$|\mathtt{out}\rangle = X^{(s_2+B_2)\cdot B_1} Z^{B_2'+B_2(1-B_1)+B_1[s_1+(s_2+B_2)B_1']} R\left(\frac{\pi}{2}\right)^{B_1} R\left(\frac{\pi}{4}\right)^{B_1'} |+\rangle$$

5.4.1 Correctness of Malicious 8-states QFactory

We prove the existence of a mapping $\mathcal{M}$ (which we will call Merge Gadget), from 2 states $|\mathtt{in}_1\rangle$ and $|\mathtt{in}_2\rangle$, where $|\mathtt{in}_1\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ and $|\mathtt{in}_2\rangle \in \{|+\rangle, |-\rangle, \left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$ to a state of the form $|\mathtt{out}\rangle = \left|+_{L\cdot\frac{\pi}{4}}\right\rangle$, where $L = L_1L_2L_3 \in \{0,1\}^3$.
Namely, as depicted in Figure 5.2, $\mathcal{M}$ is acting in the following way:

$$\mathcal{M}(|\mathtt{in}_1\rangle, |\mathtt{in}_2\rangle) = M_{X,2}M_{X,1} \wedge Z_{2,3} \wedge Z_{1,2} \left[\left|+_{\frac{\pi}{4}}\right\rangle \otimes |\mathtt{in}_1\rangle \otimes |\mathtt{in}_2\rangle\right] \tag{5.24}$$

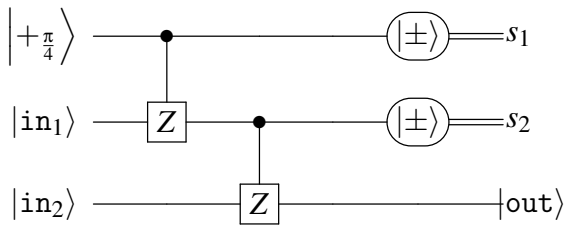


Figure 5.2: Merge Gadget

Theorem 5.4.1. *In an honest run, the Output state of the Malicious 8-states QFactory Protocol is of the form* $\left|+_{L\cdot\frac{\pi}{4}}\right\rangle$, *where* $L = L_1L_2L_3 \in \{0,1\}^3$.

Proof. In an honest run, using the Merge Gadget (Figure 5.2) we get:

$$\mathcal{M}(|\mathtt{in}_1\rangle, |\mathtt{in}_2\rangle) = M_{X,2}M_{X,1} \wedge Z_{2,3} \wedge Z_{1,2} \left[\left|+_{\frac{\pi}{4}}\right\rangle \otimes |\mathtt{in}_1\rangle \otimes |\mathtt{in}_2\rangle\right] \tag{5.25}$$

Using the correctness of Malicious 4-states QFactory (Thereom 5.2.2), we have that:

$$\begin{aligned} |\mathtt{in}_1\rangle &= H^{B_1}X^{B_2}|0\rangle \\ |\mathtt{in}_2\rangle &= R\left(\frac{\pi}{2}\right)^{B_1'} Z^{B_2'}|+\rangle \end{aligned} \tag{5.26}$$

Thus:

$$|\mathtt{out}\rangle = M_{X,2}M_{X,1}\wedge Z_{2,3}\wedge Z_{1,2}\left[\left|+_{\frac{\pi}{4}}\right\rangle \otimes H^{B_1}Z^{B_2}|+\rangle \otimes R\left(\frac{\pi}{2}\right)^{B'_1} Z^{B'_2}|+\rangle\right] \quad (5.27)$$

Which is then equivalent to:

$$|\mathtt{out}\rangle = R\left[\pi(B'_2+B_2+B_1\cdot(s_1+s_2))+\frac{\pi}{2}(B'_1+(B_2+s_2)\cdot B_1)+\frac{\pi}{4}B_1\right]|+\rangle \quad (5.28)$$

As a result, we obtain:

$$L_1 = B'_2 \oplus B_2 \oplus [B_1\cdot(s_1\oplus s_2)] \quad (5.29)$$

$$L_2 = B'_1 \oplus [(B_2\oplus s_2)\cdot B_1] \quad (5.30)$$

$$L_3 = B_1 \quad (5.31)$$

□

It can also be noticed that, in an honest run of Malicious 8-states QFactory, the Client can efficiently determine L: using b^1, b^2, y^1, y^2 and the trapdoors t_k^1, t_k^2, she first obtains (B_1, B_2) and (B'_1, B'_2), and after receiving s_1, s_2, she determines the description of the state prepared by the Server.

5.4.2 Security against Malicious Adversaries of Malicious 8-states QFactory

In any run of the protocol, honest or malicious, the state that the client believes that the server has, is given by Theorem 5.4.1.
Therefore, as in the case of Malicious 4-states QFactory, the task that a malicious server wants to achieve, is to be able to guess, as good as he can, the index of the output state that the client thinks the server has produced. In particular, in our case, the server needs to guess the bits L_2 and L_3 (corresponding to the basis) of the (honest) output state.

Definition 5.4.2 (8 states basis blindness)**.** *A protocol (π_A, π_B) achieves **basis-blindness** with respect to an ideal list of 8 states $S = \{S_{L_1,L_2,L_3}\}_{(L_1,L_2,L_3)\in\{0,1\}^3}$ if:*

- *S is the set of states that the protocol outputs, i.e.:*

$$\Pr[|\phi\rangle = S_{L_1,L_2,L_3} \in S \mid ((L_1,L_2,L_3),|\phi\rangle) \leftarrow (\pi_A \| \pi_B)] = 1$$

- *No information is leaked about the "basis" bits (L_2, L_3) of the output state of the protocol, i.e for all QPT adversary $\mathcal{A}$:*

$$\Pr\left[L_2 = \tilde{L_2} \text{ and } L_3 = \tilde{L_3} \mid ((L_1, L_2, L_3), (\tilde{L_2}, \tilde{L_3})) \leftarrow (\pi_A \| \mathcal{A})\right] \leq \frac{1}{4} + \mathsf{negl}(n)$$

Theorem 5.4.3. *Malicious 8-states QFactory satisfies 8-state basis blindness with respect to the ideal set of states* $S = \left\{\left|+_{L\cdot\frac{\pi}{4}}\right\rangle\right\}_{L\in\{0,\ldots,7\}} = \left\{|+\rangle, \left|+_{\frac{\pi}{4}}\right\rangle, .., \left|+_{\frac{7\pi}{4}}\right\rangle\right\}$.

Proof. We will prove this result by reduction showing that, if there exists a QPT adversary $\mathcal{A}$ that is able to break the 8-states basis blindness property of Malicious 8-states QFactory (determine the indices L_2 and L_3 with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$ for some polynomial function $poly_1(\cdot)$), then we can construct a QPT adversary $\mathcal{A}'$ that can break the 4-states basis blindness of the Malicious 4-states QFactory protocol (determine the basis bit with probability $\frac{1}{2} + \frac{1}{poly_2(n)}$, for some polynomial function $poly_2(\cdot)$).
The input to $\mathcal{A}'$ should be consisting only of the $\mathcal{F}$ family index, k, and the description of h. Next we show how to construct $\mathcal{A}'$ to determine the corresponding index B_1 or B_1' of the output state (of one of the 2 Malicious 4-states QFactory runs), by using as a subroutine $\mathcal{A}$ that acts as follows: receives as input 2 function indices $k^{(1)}$ and $k^{(2)}$, runs Malicious 8-states QFactory and then $\mathcal{A}$ is able to output the correct basis bits L_2 and L_3, with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$.
Before we describe $\mathcal{A}'$, we need to define the following 3 values:

- $P_2 =$ probability that $\mathcal{A}$ guesses correctly L_2;
- $P_3 =$ probability that $\mathcal{A}$ guesses correctly L_3;
- $P_\oplus =$ probability that $\mathcal{A}$ guesses correctly $L_2 \oplus L_3$;

Now, given that $\mathcal{A}$ is able to produce both L_2 and L_3 with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$, this implies that $max(P_2, P_3, P_\oplus) \geq \frac{1}{2} + \frac{1}{poly_2(n)}$ for some polynomial $poly_2(\cdot)$ (the proof is presented in section B.2).
We will construct $\mathcal{A}'$ such that if P_3 is the maximum, then $\mathcal{A}'$ can determine B_1 (break the basis blindness of the first Malicious 4-states QFactory run) and if P_2 or $P_\oplus$ is the maximum, then $\mathcal{A}'$ can determine B_1' (break the basis blindness of the the second "rotated" Malicious 4-states QFactory run).

$\mathcal{A}'(k,h,1^n)$

1 : / $k^{(1)}$ will correspond to the input for the first run of Malicious 4-states QFactory

2 : / - with the output index (B_1,B_2), while k will correspond to

3 : / the input for the second run - with the output index (B_1',B_2')

4 : $\left(k^{(1)},t_k^{(1)}\right) \leftarrow_{\$} Gen_{\mathcal{F}}(1^n)$

5 : / As the probability P of successfully guessing L_2 and L_3 is $\frac{1}{4}+\frac{1}{poly_1(n)}$

6 : / We know that $max(P_2,P_3,P_{\oplus}) \geq \frac{1}{2}+\frac{1}{poly_2(n)}$

7 : **if** $P_3 = max(P_2,P_3,P_{\oplus})$

8 : / we break the basis-blindness of the first Malicious 4-states QFactory by determining B_1

9 : $(y^{(1)},y^{(2)},b^{(1)},b^{(2)},s_1,s_2),(\tilde{L_2},\tilde{L_3}) \leftarrow \mathcal{A}(k,k^{(1)},h)$

10 : / $(y^{(1)},y^{(2)},b^{(1)},b^{(2)},s_1,s_2)$ represents the classical communication received from $\mathcal{A}$

11 : / during the run of Malicious 8-states QFactory, and

12 : / $(\tilde{L_2},\tilde{L_3})$ - are the guesses of $\mathcal{A}$ for the indices of the outcome

13 : $\tilde{B_1} \leftarrow \tilde{L_3}$

14 : **return** $\tilde{B_1}$/ as $B_1 = L_3$ as seen in Eq. 5.31 and

15 : / we have success probability $\geq \frac{1}{2}+\frac{1}{poly_2(n)}$

16 : **else** / we break the basis-blindness of the second Malicious 4-states QFactory by determining B_1'

17 : $(y^{(1)},y^{(2)},b^{(1)},b^{(2)},s_1,s_2),(\tilde{L_2},\tilde{L_3}) \leftarrow \mathcal{A}(k^{(1)},k,h)$

18 : / $(y^{(1)},y^{(2)},b^{(1)},b^{(2)},s_1,s_2)$ represents the classical communication received from $\mathcal{A}$

19 : / during the run of Malicious 8-states QFactory, and

20 : / $(\tilde{L_2},\tilde{L_3})$ - are the guesses of $\mathcal{A}$ for the indices of the outcome

21 : $(z^{(1)},z'^{(1)}) \leftarrow Inv_{\mathcal{F}}(y^{(1)},t_k^{(1)})$

22 : $B_1 \leftarrow h(z^{(1)}) \oplus h(z'^{(1)})$

23 : $B_2 \leftarrow [b_n^{(2)} + \sum(z_i^{(2)} \oplus z_i'^{(2)}) \cdot b_i^{(2)}]B_1 + h(z^{(2)})(1-B_1) \bmod 2$

24 : **if** $P_2 = max(P_2,P_3,P_{\oplus})$

25 : / Then $B_1' = L_2 \oplus B_1 \cdot (B_2 \oplus s_2)$ as seen in Eq. 5.30

26 : $\tilde{B_1'} \leftarrow \tilde{L_2} \oplus [B_1 \cdot (B_2 \oplus s_2)]$

27 : **return** $\tilde{B_1'}$

28 : **if** $P_{\oplus} = max(P_2,P_3,P_{\oplus})$

29 : $\tilde{B_1'} \leftarrow \tilde{L_2} \oplus \tilde{L_3} \oplus B_1 \oplus [B_1 \cdot (B_2 \oplus s_2)]$

30 : **return** $\tilde{B_1'}$

□

5.5 Malicious-abort 4-states QFactory: treating abort case

In this section, we will discuss an extension of Malicious 4-states QFactory, whose aim is to achieve basis blindness even against adversaries that try to exploit the fact that Malicious 4-states QFactory can abort when there is only one preimage associated to the y returned by the server. One may think that we could just send back this `accept/abort` bit to the server, but unfortunately it could leak additional information on the hardcore bit d_0 (which corresponds to the basis B_1 of the produced qubit) to the server, and as soon as the probability of acceptance is small enough, we cannot guarantee that this bit remains secret. On the other hand, for honest servers, the probability of aborting is usually non-negligible, so we cannot neglect this case.

We stress out that it is also possible to guarantee that for honest servers this probability goes negligibly close to 1 by making an appropriate choice of parameters for the function. In that case the initial protocol of Malicious QFactory defined in section 5.2 is secure, but this comes (as far as we know), at the cost of using a function which is "less" secure. More specifically, instead of having a reduction to GAPSVP with a polynomial γ, the reduction usually goes to GAPSVP with a super-polynomial γ. Such function parameters have been used implicitly in other works [Mah18] ([Bra18] later removed this assumption), and for now they are believed to be secure (the best known polynomial algorithm cannot break GAPSVP with a γ smaller than exponential), but nevertheless we aim to remove this non-standard assumption.

The solution we propose here uses the assumption that the classical Yao's XOR Lemma also applies for one-round protocols (with classical messages) against quantum adversary. This lemma roughly states that if you cannot guess the output bit of one round with probability better than η, then you cannot guess the output bit of t independents rounds with probability much better than $1/2+\eta^t$. As far as we know, this lemma has been proven only in the classical case (see [GNW11]).

In the following, we will call "accepted run" a run of Malicious 4-states QFactory such that the y received from the server has 2 preimages ("probability of success" also refers to the probability of this event when the server is honest), and otherwise we call it an "aborted run".

5.5.1 The Malicious-Abort 4-state QFactory Protocol

The solution we are proposing is to run several instances of Malicious 4-states QFactory, by remarking that we do not need to discard the aborted runs. Indeed, it is easy to see that in these cases, the produced qubits will always be in the same basis ($\{|0\rangle, |1\rangle\}$ denoted by 0). The idea is then to implement on the server side a circuit that will output a qubit having as basis the XOR of all the basis of the accepted runs (without even leaking which runs are accepted or not), and verify on client's side that the number of accepted runs is high enough (this will happen with overwhelming probability for honest servers). If it is the case, the client will just output the XOR of the basis of the accepted run, and otherwise (i.e. if the server is malicious), she will just pick a random bit value.

Unfortunately, in practice things are a bit more complicated, and in order to prove the security of our construction we need to divide all the t runs into n_c "chunks" of size t_c, and test the chunks individually. We provide now a more detailed description of the protocol and proof technique:

1. We run $t = n_c \cdot t_c$ parallel instances of Malicious 4-states QFactory, without revealing the abort bit for any of these instances;

2. The key point to note is that for honest servers, if y_i has only one preimage then the output qubit produced by the server at the end of the protocol will be either $|0\rangle$ or $|1\rangle$, but cannot be $|+\rangle$ or $|-\rangle$ (with one preimage we do not have a superposition), as we show in Lemma 5.5.2. In other words, the basis is always the $\{|0\rangle, |1\rangle\}$ basis (denoted as 0) so we do not really need to abort. Therefore, for honest runs, at the end, the basis of the output qubits will be equal to $\beta_i = d_{0,i} \cdot a_i$, for all $i \in \{1, \cdots, t\}$, where $a_i = 1$ if and only if y_i has two preimages, and $a_i = 0$ otherwise.

3. Then, from the t qubits in the basis $\beta_1, \ldots, \beta_t$, we will produce a single qubit belonging to the set $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ whose basis B_1 is the XOR of the basis of the t qubits, i.e. $B_1 = \oplus_{i=1}^{t} \beta_i$. The construction is shown in Lemma 5.5.3.

4. Then, the client will test every chunk, by checking whether the proportion of accepted runs in every chunk is greater than some parameter p_c. If all chunks have enough accepted runs, then the client just computes and outputs the real value for the basis (which is the XOR of the hardcore bits of all the accepted runs) and the value bit B_2. However, if at least one chunk does not have enough

accepted runs (which cannot happen if the server is honest), then the client just outputs random values for the basis and value bit, not correlated with server's qubit (equivalent to saying that a malicious server can always throw the qubit and pick a new qubit, not correlated with client's state description).

5. Correctness: if the probability of $\mathcal{F}$ to have two preimages (given an honest server) is at least a constant p_a greater than $1/2$ (the parameters we propose in Lemma 5.3.3 guarantee this property), and if t is chosen high enough, the fraction of accepted runs will be close to p_a, and we can show that the probability to have a fraction of accepted runs smaller than a given constant $p_b < p_a$ is exponentially (in t) close to 0 (cf Lemma 5.5.4). So with overwhelming probability, all the chunks will have enough accepted runs, i.e. honest servers will have a qubit corresponding to the output of the client.

6. Soundness: to prove the security of this scheme, we first prove in Lemma 5.5.7 that it is impossible for any adversary to guess the output of one chunk with a probability bigger than a constant $\eta < 1$ (otherwise we have a direct reduction that breaks the hardcore bit property of g_K). Now, using the quantum version of Yao's XOR Lemma that we conjecture at Conjecture 5.5.1, we can deduce that no malicious server is able to guess the XOR of the t_c chunks/instances with probability better than $1/2 + \eta^{t_c} + \mathsf{negl}(n)$, which goes negligibly close to $1/2$ when $t_c = \Omega(n)$.

Putting everything together, the parties will run $t = n_c \cdot t_c$ Malicious 4-states QFactory in parallel, the client will then check if $\sum_i a_i$ is higher than $p_c \cdot t_c$ for all the n_c chunks, and if so, she will set $B_1 = \oplus_{i=1}^{t} d_i \cdot a_i$ (server has a circuit to produce a qubit in this basis as well). Otherwise B_1 will be set to a uniformly chosen random bit (it is equivalent to say that a malicious server can destroy the qubit, and this is also unavoidable even with a real quantum communication), and we still have correctness with overwhelming probability for honest clients. The exact algorithm is described in Protocol 5, while the security result is shown in Theorem 5.5.8.

5.5.2 Correctness and Security Malicious-Abort 4-state QFactory

In this section we will formalize and prove the previous statements.

Conjecture 5.5.1 (Yao's XOR Lemma for one-round protocols (with classical messages) against quantum adversary)**.**

Let n be the security parameter, let $f_n : \mathcal{X}_n \times \mathcal{Y}_n \to \{0,1\}$ *be a (possibly non-deterministic) family of functions (usually not computable in polynomial time), and let* χ_n *be a distribution on* $\mathcal{X}_n$ *efficiently samplable. If there exists* $\delta(n)$ *such that* $|\delta(n)| \geq \frac{1}{\mathsf{poly}(n)}$ *and such that for all polynomial (in n) quantum adversary* $\mathcal{A}_n : \mathcal{X}_n \to \mathcal{Y}_n \times \{0,1\}$,

$$\Pr\left[\tilde{\beta} = f_n(x,y) \mid (y,\tilde{\beta}) \leftarrow \mathcal{A}_n(x), x \leftarrow \chi_n\right] \leq 1 - \delta(n)$$

then, for all $t \in \mathbb{N}^*$ *and for all QPT adversary* $\mathcal{A}'_n : \mathcal{X}_n^t \to \mathcal{Y}_n^t \times \{0,1\}$, *we have:*

$$\begin{aligned}&\Pr\left[\tilde{\beta} = \bigoplus_{i=1}^{t} f_n(x_i,y_i) \mid (y_1,\ldots,y_t,\tilde{\beta}) \leftarrow \mathcal{A}'_n(x_1,\ldots,x_t), \forall i, x_i \leftarrow \chi_n\right]\\ \leq &\frac{1}{2} + (1-\delta(n))^t + \mathsf{negl}(n)\end{aligned}$$

Lemma 5.5.2 (Aborted runs are useful)**.** *If Client (*π_{A_4}*) and Server (*π_{B_4}*) are following the Malicious 4-states QFactory protocol honestly, and if y does not have not 2 preimages, then the output qubit produced by* π_{B_4} *is in the basis* $\{|0\rangle, |1\rangle\}$.

The proof can be found in section B.3.

Lemma 5.5.3 (Gadget circuit $\mathtt{Gad}_\oplus$ computes XOR)**.** *If we denote by* b_i *the basis of* $|\mathtt{in}_i\rangle$ *(equal to* 0 *if the basis is* $0/1$*, and 1 if the basis is* $+/-$*), then by running the circuit* $\mathtt{Gad}_\oplus$ *represented in Figure 5.3 on these inputs, then the basis of the output of the circuit,* $|\mathtt{out}\rangle$ *is equal to* $\oplus_{i=1}^{t} b_i$.

The proof can be found in section B.3.

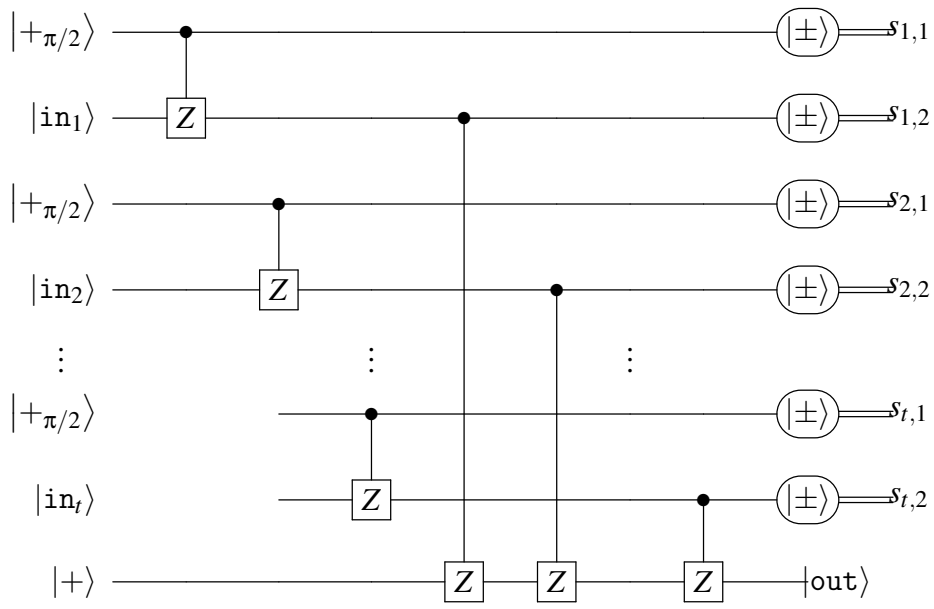


Figure 5.3: The XOR gadget circuit $\mathtt{Gad}_\oplus$ (performed by Server)

We can now describe the protocol of Malicious-Abort 4-states QFactory:

Protocol 5 Malicious-Abort 4-states QFactory Protocol

Requirements:

Public: The family of functions $\mathcal{F}$ and h, such that the probability of having two preimages for a random image is greater than a constant $p_a > 1/2$.

Parameters:

- $n_c \in \mathbb{N}$ - number of chunks;
- $t_c \in \mathbb{N}$ - number of repetitions per chunk;
- $p_a \in (1/2, 1]$ - lower bound on probability of accepted run in the honest protocol;
- $p_c \in (1/2, 1] < p_a$ - threshold on fraction of accepted runs per chunk;

Parameters can be chosen in such a way to have overwhelming probability of success for honest players, and negligible advantage for an adversary trying to guess the basis.

Stage 1: Run multiple QFactories

– Client: prepares $t = n_c \cdot t_c$ public keys and trapdoors:

$$\left((k^{(i,j)}, t_{k^{(i,j)}}) \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^n)\right)_{i\in\{1,\cdots,n_c\}, j\in\{1,\cdots,t_c\}}$$

The Client then sends the public keys $k^{(i,j)}$ to the Server, together with h.

– Server and Client: follow Protocol 3 for t times, with the keys sent at the step before. Client receives $((y^{(i,j)}, b^{(i,j)}))_{i,j}$, and computes for all i, j:

$$a^{(i,j)} := \begin{cases} 1, & \text{if } |f^{-1}(y^{(i,j)})| = 2 \\ 0, & \text{otherwise} \end{cases} \tag{5.32}$$

- If $a^{(i,j)} = 1$ Client computes $B_1^{(i,j)}$ and $B_2^{(i,j)}$ as in Protocol 3;
- Else if $a^{(i,j)} = 0$, Client sets $B_1^{(i,j)} = 0$ and $B_2^{(i,j)} = h(f^{-1}(y))$;
- Server obtains t outputs $\left|\mathtt{in}_{(i,j)}\right\rangle$.

Stage 2: Combine runs and output

– Server: applies circuit $\mathtt{Gad}_{\oplus}$ (Figure 5.3) on the t outputs $|\mathtt{in}_t\rangle$, and outputs $|\mathtt{out}\rangle$.

– Client: checks that for every chunks $i \in \{1, n_c\}$ the number of accepted runs is high enough, i.e.: $\sum_{j=1}^{t_c} a^{(i,j)} \geq p_c t_c$.

- If at least one chunk does not respect this condition, then Client picks two random bits B_1 (the basis bit) and B_2 (the value bit) and outputs (B_1, B_2), corresponding to the description of the BB84 state $H^{B_1} X^{B_2} |0\rangle$.
- If all n_c chunks respect this condition, then Client sets $B_1 := \bigoplus_{i,j} B_1^{(i,j)}$ (the final basis is the XOR of all the basis), and B_2 will be chosen to match the output of $\mathtt{Gad}_{\oplus}$ (Figure 5.3).

Lemma 5.5.4 (Probability of correctness of Malicious-Abort 4-states QFactory for one chunk)**.** *If the probability to have an accepted run in Malicious 4-states QFactory with honest parties is greater than a constant $p_a > 1/2$, i.e.:*

$$\Pr\left[|f_k^{-1}(y)| = 2 \mid (\pi_{A_4} \| \pi_{B_4})\right] \geq p_a$$

(where π_{A_4} and π_{B_4} are the honest protocols of Malicious 4-states QFactory) then for any constant $p_b < p_a$, the probability to have in a chunk of t_c runs, at least $p_b t_c$ accepted runs is exponentially in t_c close to 1:

$$\Pr\left[\sum_{i=1}^{t_c} a_i \geq p_b t_c \mid (\pi_{A_4 \oplus c}^{t_c} \| \pi_{B_4 \oplus c}^{t_c})\right] \geq 1 - \frac{1}{e^{2(p_a - p_b)^2 t_c}} = 1 - \mathsf{negl}(t_c)$$

(where $\pi_{A_4 \oplus c}^{t_c}$ and $\pi_{B_4 \oplus c}^{t_c}$ are the (honest) parties of the Protocol 5 restricted on one chunk of size t_c, or, equivalently t_c parallel repetitions of Protocol 3)

The proof can be found in section B.3.

Lemma 5.5.5 (Correctness of Protocol 5)**.** *Protocol 5 is correct with overwhelming probability as soon as $t = \mathsf{poly}(n)$ and $t_c = \Omega(n)$, i.e.*

$$\Pr\left[|\mathtt{out}\rangle = H^{B_1} Z^{B_2} \mid ((B_1, B_2), |\mathtt{out}\rangle) \leftarrow (\pi_A \| \pi_B)\right] \geq 1 - \mathsf{negl}(n)$$

The proof can be found in section B.3.

Definition 5.5.6. *For any public key k and image y, we define:*

$$a(k,y) := \begin{cases} 1, & \text{if } |f_k^{-1}(y)| = 2 \\ 0, & \text{otherwise} \end{cases} \tag{5.33}$$

Then, for all $t_c \in \mathbb{N}$ and $p_c \in [0,1]$, we define the randomized function:

$$\beta_{t_c, p_c}\left(k^{(1)}, \ldots, k^{(t_c)}, y^{(1)}, \ldots, y^{(t_c)}\right) := \begin{cases} B = \oplus_{i=1}^{t_c} a\left(k^{(i)}, y^{(i)}\right) \cdot d_0^{(i)} & , \text{ if } \sum_{i=1}^{t_c} a(k^{(i)}, y^{(i)}) \geq p_c \cdot t_c \\ u & , \text{ if } \sum_{i=1}^{t_c} a(k^{(i)}, y^{(i)}) < p_c \cdot t_c \end{cases} \tag{5.34}$$

where u is a uniform random bit and $d_0^{(i)}$ is the hardcore bit corresponding to $k^{(i)} := (K^{(i)}, g_{K^{(i)}}(z_0^{(i)}))$, i.e. $d_0^{(i)} = h(z_0^{(i)})$.

Lemma 5.5.7 (Solving one chunk is difficult)**.** *Let $p_c \in (\frac{1}{2}, 1]$. Then, for all QPT adversary $\mathcal{A}$ we have:*

$$\Pr\Big[\tilde{B}_1 = \beta_{t_c,p_c}\left(k^{(1)},\dots,k^{(t_c)},y^{(1)},\dots,y^{(t_c)}\right) \mid \left(y^{(1)},\dots,y^{(t_c)},\tilde{B}_1\right) \leftarrow \mathcal{A}\left(k^{(1)},\dots,k^{(t_c)}\right)\Big] < \eta$$

with $\eta = \frac{1}{2}\left(1+\frac{1}{2p_c}\right)$, where the randomness is over the randomness of β, $\mathcal{A}$, and over the choices of $(k^{(i)})_i$ and $(y^{(i)})_i$.

Proof. By contradiction, let us assume that there exists a QPT adversary $\mathcal{A}$ outputting $\tilde{B}_1$ and $\left((y^{(1)},\dots,y^{(t_c)}\right)$ such that:

$$\Pr\left[\tilde{B}_1 = \beta\right] > \eta$$

where we omitted the parameters for readability.
We can rewrite $\Pr\left[\tilde{B}_1 = \beta\right]$ as:

$$\begin{aligned}\Pr\left[\tilde{B}_1 = \beta\right] = \Pr\left[\tilde{B}_1 = \beta \mid \sum_{i=1}^{t_c} a(k^{(i)},y^{(i)}) \geq p_c \cdot t_c\right] \cdot \Pr\left[\sum_{i=1}^{t_c} a(k^{(i)},y^{(i)}) \geq p_c \cdot t_c\right] + \\ + \Pr\left[\tilde{B}_1 = \beta \mid \sum_{i=1}^{t_c} a(k^{(i)},y^{(i)}) \leq p_c \cdot t_c\right] \cdot \Pr\left[\sum_{i=1}^{t_c} a(k^{(i)},y^{(i)}) \leq p_c \cdot t_c\right]\end{aligned} \tag{5.35}$$

Now, using the definition of the function β and by using the notation $\alpha := \Pr\left[\sum_{i=1}^{t_c} a(k^{(i)},y^{(i)}) \leq p_c \cdot t_c\right]$, we have:

$$\begin{aligned}\Pr\left[\tilde{B}_1 = \beta\right] &= \Pr\left[\tilde{B}_1 = \oplus_{i=1}^{t_c} a\left(k^{(i)},y^{(i)}\right) \cdot d_0^{(i)}\right] \cdot (1-\alpha) + \Pr\left[\tilde{B}_1 = u\right] \cdot \alpha \\ &= \Pr\left[\tilde{B}_1 = \oplus_{i=1}^{t_c} a\left(k^{(i)},y^{(i)}\right) \cdot d_0^{(i)}\right] \cdot (1-\alpha) + \frac{\alpha}{2} \leq 1-\alpha+\frac{\alpha}{2} = 1-\frac{\alpha}{2}\end{aligned} \tag{5.36}$$

Therefore, we also have: $\alpha < 2(1-\eta)$.
Now, let us examine $\Pr\left[\tilde{B}_1 = B\right]$, the probability that the adversary $\mathcal{A}$ can output $B = \oplus_{i=1}^{t_c} a\left(k^{(i)},y^{(i)}\right) \cdot d_0^{(i)}$.
We will show through a reduction that if $\mathcal{A}$ can output B with probability p, then there exists a QPT adversary $\mathcal{A}'$ that can break the hardcore predicate property of g_k (determine the hardcore predicate d_0 associated with k) with probability at least $(1-\alpha)p_c \cdot p$.

$\mathcal{A}'(k)$

Runs $t_c - 1$ times $Gen_{\mathcal{F}}$ obtaining $\{(k^{(i)}, t_{k^{(i)}})\}_{i=1}^{t_c-1}$
Calls $\mathcal{A}(k, k^{(1)}, \cdots, k^{(t_c-1)}) \rightarrow (\tilde{B}_1, \{y, y^{(1)}, \cdots y^{(t_c-1)}\})$
for $i = 1, \cdots t_c - 1$:
    Compute $d_0^{(i)} \leftarrow d_0(t_{k^{(i)}})$ // As indicated in Protocol 3
    Run $Inv_{\mathcal{F}}(y^{(i)}, t_{k^{(i)}})$
    if $y^{(i)}$ has 2 preimages **then**
        $a(k^{(i)}, y^{(i)}) \leftarrow 1$
    else $a(k^{(i)}, y^{(i)}) \leftarrow 0$
$\tilde{d}_0 \leftarrow \oplus_{i=1}^{t_c-1} a(k^{(i)}, y^{(i)}) \cdot d_0^{(i)} \oplus \tilde{B}_1$ // this will represent the guess of $\mathcal{A}'$ for $d_0(t_k)$ - the hp corresponding to g_k
return $\tilde{d}_0$

We can see that in order for $\mathcal{A}'$ to output the correct d_0 hardcore predicate of g_k, 3 things must happen: 1) $\mathcal{A}$ to output the correct B, 2) $\sum_{i=1}^{t_c} a(k^{(i)}, y^{(i)}) \geq p_c \cdot t_c$ and 3) the y outputted by $\mathcal{A}$ corresponding to function k has 2 preimages (and hence $a(k, y) = 1$). Then, we compute the probability of success for $\mathcal{A}'$ as:

$$\Pr\left[\mathcal{A}'(k) = d_0\right] = p \cdot \Pr\left[\sum_{i=1}^{t_c} a(k^{(i)}, y^{(i)}) \geq p_c \cdot t_c\right] \cdot p_a \geq p \cdot (1 - \alpha) \cdot p_c \tag{5.37}$$

Therefore, as d_0 is a hardcore predicate we deduce:

$$\Pr\left[\tilde{B}_1 = \oplus_{i=1}^{t_c} a\left(k^{(i)}, y^{(i)}\right) \cdot d_0^{(i)}\right] \cdot (1 - \alpha) \cdot p_c \leq \frac{1}{2} + \mathsf{negl}(n) \tag{5.38}$$

Therefore, by returning to Eq.(5.36), we have:

$$\begin{aligned} \Pr\left[\tilde{B}_1 = \beta\right] &= \Pr\left[\tilde{B}_1 = \oplus_{i=1}^{t_c} a\left(k^{(i)}, y^{(i)}\right) \cdot d_0^{(i)}\right] \cdot (1 - \alpha) + \frac{\alpha}{2} \\ &\leq \frac{1}{2p_c} + \frac{\alpha}{2} + \mathsf{negl}(n) \\ &\leq \frac{1}{2p_c} + 1 - \eta + \mathsf{negl}(n) \end{aligned} \tag{5.39}$$

From our assumption we have that $\Pr\left[\tilde{B}_1 = \beta\right] > \eta$, hence this gives us:

$$\eta < \frac{1}{2} + \frac{1}{4p_c} + \mathsf{negl}(n) \tag{5.40}$$

Because η and p_c are constants that do not depend on n, this equality is also true without the $\mathsf{negl}(n)$ term:

$$\eta < \frac{1}{2} + \frac{1}{4p_c} \tag{5.41}$$

which contradicts that $\eta = \frac{1}{2} + \frac{1}{4p_c}$ and completes our proof. □

Finally, by combining all these results we obtain the correctness and security of the full Malicious-Abort QFactory (Protocol 5).

Theorem 5.5.8 (Malicious-Abort QFactory is correct and secure)**.** *Assuming Conjecture 5.5.1, and by ensuring that the probability of the family $\mathcal{F}$ to have two preimages for any image is bigger than a constant $p_a > 1/2$, then there exists a set of parameters p_c, t_c and n_c such that Protocol 5 is correct with probability exponentially close to* 1 *and basis-blind, i.e. for any QPT adversary $\mathcal{A}$:*

$$\Pr\left[\tilde{B}_1 = B_1 \mid ((B_1,B_2),\tilde{B}_1) \leftarrow (\pi_{A_4\oplus}\|\mathcal{A})\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

More precisely, we need $t_c \in (1/2, p_c)$ to be a constant, and both t_c and n_c need to be polynomial in n and $\Omega(n)$.

Proof. Firstly, the overwhelming probability of correctness is ensured by Lemma 5.5.5. For security, using Lemma 5.5.7, we know that there exists a constant $\eta < 1$ such that no adversary can solve a chunk (compute β_{t_c,p_c}) with probability better than η.
Now, we will use Conjecture 5.5.1 in the following way. The function f_n will be equal to β_{t_c,p_c} defined in Definition 5.5.6 (for each chunk). Then the input x sampled efficiently from χ_n provided to the adversary is represented by the set of t_c public keys $\{k^{(i)}\}_{i=1}^{t_c}$ and the outputs y given by $\mathcal{A}$ is the set of images $\{y^{(i)}\}_{i=1}^{t_c}$. Then, we need to ensure that $\delta(n) = 1-\eta$ is at least $\frac{1}{poly(n)}$, which holds as in our case η is a constant ($\eta = \frac{1}{2} + \frac{1}{4p_c}$).
Then Conjecture 5.5.1 implies that for any number of chunks n_c, no QPT adversary can get the XOR of the solutions of n_c chunks with probability better than $\frac{1}{2} + \eta^{n_c} + \mathsf{negl}(n)$. As η is a constant and $n_c = poly(n)$ we have that:
No QPT adversary can obtain the basis B_1 of Protocol 5 with probability better than $\frac{1}{2} + \mathsf{negl}(n)$, which concludes our proof.

□

Chapter 6

Security Limitations of Classical Client Delegated Quantum Computing

One of the central building blocks to achieve secure delegation of quantum computations, as well as other client-server functionalities is secure remote state preparation (RSP) defined first in [DKL11]. The RSP resources enable the Client to remotely prepare a quantum state on Server's side and as a result, they are the natural candidate to replace quantum channel resources in a modular fashion. These resources further appear to enable a large family of composable protocols [DKL11, DFPR14], including the Universal Blind Quantum Computation (UBQC) protocol [BFK09] used to delegate a computation to a remote quantum Server who learns no information about the ongoing computation.

However, secure delegated quantum computing is typically achieved in the settings when both Client and Server have access to quantum resources, in particular via quantum communication such that the Client can establish the necessary correlations with the Server to securely perform this task. As a result, the question becomes relevant whether it is possible to rely solely on classical channels between Client and Server and still benefit from its quantum capabilities while preserving the same security level.

Motivated by this, we first introduced in chapter 4 the protocol $\mathsf{HBC-QFactory}$ mimicking the remote state preparation resource over a purely classical channel, enabling a fully classical Client (using exclusively classical communication resources) to remotely prepare a quantum state on Server's side, under the assumption that the Learning-With-Errors problem is hard to solve for quantum computers. A further example from the family of classical-client remote state preparation protocols, denoted in this chapter by $\mathsf{RSP}_{\mathsf{CC}}$, is the Malicious 4-states QFactory described in chapter 5,

with enhanced security compared to $\mathsf{HBC-QFactory}$.

The important role of such classical RSP primitives as part of larger protocols, most notably in their role of replacing quantum channels between Client and Server, is emphasized by their ability to allow classical-server functionalities, such as delegated quantum computing, available to classical users. Therefore, it is of utmost importance to develop an understanding of this primitive, notably its security guarantees when composed in larger contexts, such as in [GV19].

In this chapter we will investigate the *composable security* of classical client remote state preparation. More specifically, we study what is the privacy loss when employing $\mathsf{RSP}_{\mathsf{CC}}$ as a *sub-module*, and we address this question using the Constructive Cryptography framework of Maurer and Renner [MR11] to provide a clear analysis of the RSP resources from a composable perspective. To begin with, we define the goal of $\mathsf{RSP}_{\mathsf{CC}}$ as the construction of ideal RSP resources from classical channels and most importantly, we reveal the security limitations of using $\mathsf{RSP}_{\mathsf{CC}}$ in general, but also in specific contexts.

Firstly, we determine a fundamental relation between the construction of ideal RSP resources from classical channels and the task of cloning quantum states with auxiliary information. We will prove that any classically constructed RSP resource must leak the full description (possibly in an encoded form) of the generated quantum state, even when we target *computational security* only. As a consequence, we find that the realization of common RSP resources, without weakening their security guarantees drastically, is impossible due to the no-cloning theorem.

Secondly, this result does not rule out that a specific $\mathsf{RSP}_{\mathsf{CC}}$ protocol can replace the quantum channel at least in some contexts, such as the Universal Blind Quantum Computation protocol. However, we show that the resulting classical client UBQC protocol cannot maintain its proven composable security as soon as $\mathsf{RSP}_{\mathsf{CC}}$ is used as a subroutine.

6.1 Overview of Contributions and Proof Techniques

In this chapter, we will cover the security of $\mathsf{RSP}_{\mathsf{CC}}$, the class of remote state preparation protocols which use only a classical channel and the use-case that corresponds to its arguably most important application: classical client delegated quantum computation achieved through the UBQC protocol with a completely classical Client. More specifically, we analyze the security of $\mathsf{UBQC}_{\mathsf{CC}}$, the family of protocols where an

$\mathsf{RSP}_{\mathsf{CC}}$ is used in order to replace the quantum channel required for the original quantum client UBQC protocol.

An example of an RSP resource is the $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ resource depicted in Figure 6.1 (where $\mathbb{Z}_{\frac{\pi}{2}}$ refers to the set of the 4 angles $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$). This resource outputs the quantum state $|+_\theta\rangle$ on its right interface, and the classical description of this state, θ, on its left interface.

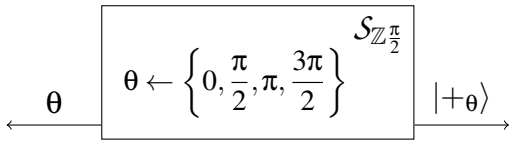


Figure 6.1: Ideal resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$

We will show in 6.2 a wide-ranging limitation of the composable guarantees that any protocol in the family of protocols $\mathsf{RSP}_{\mathsf{CC}}$ can achieve. This limitation follows from the relation between: i) the notion of classical realization of the RSP resource and ii) the notion of describability, which is a property of resources that intuitively measures how leaky is an RSP. This relation directly affects the amount of additional leakage on the classical description of the quantum state. In this way, it rules out a wide set of desirable resources, even against *computationally bounded* distinguishers.

Theorem 6.2.6 (Security Limitations of $\mathsf{RSP}_{\mathsf{CC}}$). *Any* RSP *resource, realizable by an* $\mathsf{RSP}_{\mathsf{CC}}$ *protocol with security against quantum polynomial-time distinguishers, must leak an encoded, but complete description of the generated quantum state to the server.*

The importance of Theorem 6.2.6 lies in the fact that it is drawing a connection between a *computational* notion - the composability of an $\mathsf{RSP}_{\mathsf{CC}}$ protocol with an *information theoretic* notion - the statistical leakage of the ideal resource it is constructing. This allows us to use fundamental physical principles such as *no-cloning* and *no-signalling* in the security analysis of computationally secure $\mathsf{RSP}_{\mathsf{CC}}$ protocols.

As one direct application of this powerful tool, we show that a computationally secure implementation of the ideal resource in Figure 6.1 would give rise to the construction of a quantum cloner and is hence impossible.

Proof sketch. While Theorem 6.2.6 applies to much more general RSP resources having arbitrary behaviour at its interfaces and targeting any output quantum state, for simplicity and clearance we will exemplify the main ideas of the proof of Theorem 6.2.6

for the underlying resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$.

The composable security of a protocol realizing $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ implies by definition, the existence of an efficient (BQP) simulator σ, which turns the right interface of the ideal resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ (outputting $|+_\theta\rangle$ into a completely classical interface (given that the communication between Client and Server must be entirely classical), as depicted in Figure 6.2.

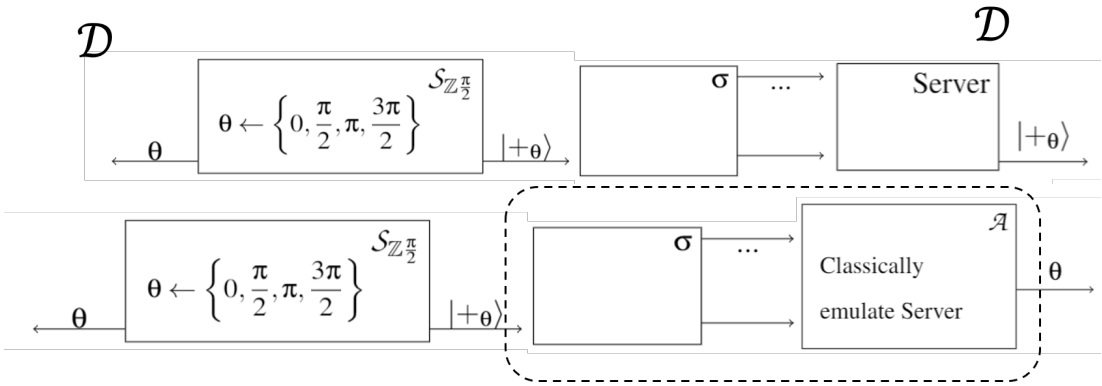


Figure 6.2: $\mathcal{D}$ denotes the polynomial-time distinguisher having access to the left interface of $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ and to the classical messages sent by σ. $\mathcal{D}$ will run the honest Server using the transcript from σ. $\mathcal{A}$ is the exponential-time algorithm that runs the same computations as the honest Server by emulating him. In this way, the classical description θ can be extracted, resulting in the algorithm in the dashed part representing the quantum cloner.

The *polynomially-bounded* quantum distinguisher will have access to this classical interface as well as the left interface of $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ and will run the protocol of the honest server, which will allow him to reconstruct the quantum state $|+_\theta\rangle$ received by the simulator from the ideal resource. Since the distinguisher has access to θ via the left interface, he can simply perform a quantum measurement to verify that the state obtained after interacting with the simulator corresponds to the classical description θ. By the correctness of the protocol, the quantum state obtained by the distinguisher, $|+_\theta\rangle$, must therefore comply with θ.

Now, since the quantum state $|+_\theta\rangle$ is transmitted from σ to the distinguisher over a classical channel, the ensemble of exchanged classical messages must contain a complete encoding of the description of the state, namely θ. A (possibly unbounded) algorithm can hence extract the actual description of the state by means of a classical emulation of the honest Server, as shown in Figure 6.2. This property of the ideal resource of being able to extract the description of the underlying quantum state is

central to our proof technique and we will call it *describability*.

Finally, having a full description of the quantum state produced by $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ would allow to create several copies of this state, a procedure prohibited by the no-cloning theorem. Consequently, we conclude that the resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ cannot be constructed from a classical channel only. □

One could attempt to modify the RSP ideal resource to incorporate some leakage to the Server about the classical description of the state, which would be necessary as the above result shows. However, this yields an ideal resource that is not a useful idealization or abstraction of the real world which puts in question whether they are at all useful in a composable analysis.

Consider for example constructions of composite protocols that utilize the (non-leaky) ideal resource as a sub-module. These constructions require a fresh security analysis if the sub-module is replaced by any leaky version of it, but since the modified resource is very specific and must mimic its implementation (in terms of leakage) it appears that this replacement does not give any benefit compared to directly using the implementation as a subroutine and then examining the composable security of the combined protocol as a whole. This latter way is therefore examined next.

More precisely, we might still be able to use $\mathsf{RSP}_{\mathsf{CC}}$ as a subroutine in other specific protocols and expect the overall protocol to still construct a useful ideal functionality. The family of protocols $\mathsf{UBQC}_{\mathsf{CC}}$ is such an application. Unfortunately, as we show in section 6.3, $\mathsf{UBQC}_{\mathsf{CC}}$ fails to provide the expected composable security guarantees once classical remote state preparation protocols are used to replace the quantum channel between Client and Server [1]. This result holds even if the distinguisher is computationally bounded.

Theorem 6.3.10 (Impossibility of $\mathsf{UBQC}_{\mathsf{CC}}$). *No* $\mathsf{RSP}_{\mathsf{CC}}$ *protocol can replace the quantum channel in the* UBQC *protocol while preserving composable security.*

Proof sketch. We first show that the existence of any composable $\mathsf{UBQC}_{\mathsf{CC}}$ protocol, in the sense of achieving the ideal UBQC resource, implies the existence of a composable single-qubit $\mathsf{UBQC}_{\mathsf{CC}}$ protocol. Then, the impossibility of composable single-qubit $\mathsf{UBQC}_{\mathsf{CC}}$ protocols is shown in 2 steps. Firstly, we show that single-qubit $\mathsf{UBQC}_{\mathsf{CC}}$

[1] By composable security of UBQC we refer to the goal of achieving the established ideal functionality of [DFPR14], which we recall in section 6.3.

can be turned into RSP protocols. As a result, this allows us to employ the toolbox we developed before on RSP protocols. As a second step, we deduce that an RSP protocol of this specific kind (leaking the classical description of the underlying quantum state in the form of an encoded message) would imply a violation of the no-signaling principle, therefore showing that a composable $\mathsf{UBQC}_{\mathsf{CC}}$ protocol could not have existed in the first place. □

Before presenting our main results outlined above, we need to first introduce some notation used throughout the following sections.

6.1.1 Notations

We will denote by $\mathbb{Z}_{\frac{\pi}{2}}$ the set of the 4 angles $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, and $\mathbb{Z}_{\frac{\pi}{4}} = \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}$ the similar set of 8 angles. If ρ is a quantum state, $[\rho]$ will represent the *classical* representation (as a density matrix) of this state. We also denote the quantum state $|+_\theta\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$, where $\theta \in \mathbb{Z}_{\frac{\pi}{4}}$, and for any angle θ, $[\theta]$ will denote $[|+_\theta\rangle\langle+_\theta|]$, i.e. the classical description of the density matrix corresponding to $|+_\theta\rangle$. For a protocol $\mathcal{P} = (P_1, P_2)$ with two interacting algorithms P_1 and P_2 denoting the two participating parties, let $r \leftarrow \langle P_1, P_2\rangle$ denote the execution of the two algorithms, exchanging messages, with output r. We use the notation $\mathcal{C}$ to denote the *classical channel* resource, that just forwards classical messages between the two parties.

6.2 Impossibility of Composable Classical RSP

In this section we proceed by defining the general notion of RSP, what it tries to achieve in terms of resources and then we will quantify the information that an ideal RSP resource must leak (at its interface) to the server even when the distinguisher is computationally bounded. One would expect that against bounded distinguishers, the resource RSP can express clear privacy guarantees, which we prove cannot be the case.

The reason can be intuitively summarized as follows: assuming that there exists a simulator making the ideal resource indistinguishable from the real protocol, we can exploit this fact to construct an algorithm that can classically describe the quantum state given by the ideal resource. It is not difficult to see that there would exist an inefficient algorithm (i.e. running in exponential time) achieving this task. We show that even a computationally bounded distinguisher can distinguish the real protocol from

the ideal protocol whenever the simulator's strategy is independent of the classical description of the quantum state sent by the ideal resource. This implies that for an RSP protocol to be composably secure, there must exist a simulator that possesses at least a classical transcript encoding the description of the underlying quantum state. This fact coupled with the quantum no-cloning theorem implies that the most meaningful and natural RSP resources cannot be realized from a classical channel alone. We conclude the section by examining the class of imperfect (describable) RSP resources which avoid the no-go result at the price of being "fully-leaky", not standard, and having an unfortunately unclear composable security.

6.2.1 Remote State Preparation and Describable Resources

We first introduce, based on the standard definitions of the Constructive Cryptography framework, the notions of *correctness* and *security* for a two-party protocol between an honest Client and a malicious Server, which constructs (realizes) a resource from a classical channel resource $\mathcal{C}$.

Definition 6.2.1 (Classically-Realizable Resource)**.** *An ideal resource $\mathcal{S}$ is said to be* ε-classically-realizable *if it is realizable (in the sense of Definition 2.5.1) from a* classical *channel, i.e. if there exists a protocol $\pi = (\pi_A, \pi_B)$ between two parties (interacting classically) such that:*

$$\mathcal{C} \xrightarrow[\varepsilon]{\pi} \mathcal{S} \tag{6.1}$$

A simple ideal prototype capturing the goal of RSP can be phrased as follows: the resource outputs a quantum state (chosen at random from a fixed set of states) on one interface to the Server and the classical description of that state on the other interface to the Client. However, for our purposes this view is too narrow and we would like to generalize this definition of the resource. For instance, a resource could accept some input from the Client or could interact with the server and still be powerful enough to comply with the above described basic behaviour, when both Client and Server follow the protocol.

For this reason, we would like to capture than any resource can be categorized as an RSP resource as soon as there exists a way to efficiently convert the Client and Server interfaces to comply with the basic prototype. To formalize, this we would need to introduce the following converters that will ensure this:

1. Converter $\mathcal{A}$ will output (to the Client), after interacting with the ideal resource[2], a classical description $[\rho]$ which is one of the following:

   (a) A density matrix corresponding to a quantum state ρ.

   (b) The null matrix, in order to denote the fact that we detected some deviation that should not happen in an honest run.

2. A converter $\mathcal{Q}$, whose goal is to output (to the Server) a quantum state ρ' as close as possible to the state ρ output by $\mathcal{A}$.

3. A converter $\mathcal{P}$, whose goal is to output a classical description $[\rho']$ of a quantum state ρ' which is on average "close"[3] to ρ.

An RSP must meet 2 central criteria:

1. *Accuracy* of the classical description of the obtained quantum state. More specifically, we require that the quantum state ρ described by $\mathcal{A}$'s output to be close to $\mathcal{Q}$'s output ρ', in terms of trace distance.

2. *Purity* of obtained quantum state. Since the RSP resource aims to replace a noise-free quantum channel, it is desirable that the quantum state output by $\mathcal{Q}$ to admit a high degree of purity, i.e. that trace of ρ'^2 to be close to 1. Since ρ' is required to be close to ρ, this implies a high purity of ρ as well.

These 2 conditions on the states ρ and ρ' can be unified and equivalently captured by requiring that the quantity $Tr(\rho\rho')$ to be close to 1 as shown in Lemma C.1.3.

We can also gain a more information theoretical intuition of RSP by considering that an RSP resource together with the converters $\mathcal{A}$ and $\mathcal{Q}$) can be understood not only as a box that produces a quantum state together with its description, but also a box who accuracy can be easily and precisely *tested*. For example, if such a box produces a state ρ' and claims that the description of that state corresponds to a quantum state $|\phi\rangle$ (i.e. $[\rho] = [|\phi\rangle\langle\phi|]$), then the natural way to test the box, would be to measure ρ' by doing a projection on $|\phi\rangle$. This test will pass with probability $p_s = \langle\phi|\,\rho'\,|\phi\rangle$, and thus if the box is perfectly accurate (i.e. if $\rho' = |\phi\rangle\langle\phi|$), the test will always succeed. On the other hand,

[2] $\mathcal{A}$ is allowed to interact with the ideal resource in a non-trivial manner. However, $\mathcal{A}$ will often be the trivial converter in the sense that it simply forwards the output of the ideal resource to the Client, or – when the resource waits for a simple activation input – picks some admissible value as input to the ideal resource and forwards the obtained description to its outer interface.

[3] The closeness is defined in the same way in the two cases corresponding to $\mathcal{Q}$ and $\mathcal{P}$

when ρ' is far from the state $|\phi\rangle\langle\phi|$, this test is very unlikely to pass and we will have p_s much smaller than 1. We can then generalise this testing for arbitrary (eventually non-pure) states [4] by remarking that $p_s = \langle\phi|\rho'|\phi\rangle = Tr(|\phi\rangle\langle\phi|\rho') = Tr(\rho\rho')$. This last expression corresponds to [5] the probability of outputting 0 when measuring the state ρ' according to the POVM described by the measurement operators $E_0 := \rho$ and $E_1 = I - \rho$. Since the classical description of ρ is known, then it is also possible to perform this POVM and test the average accuracy of our box. This further intuition motivates the following definition for general RSP resources.

Definition 6.2.2 (RSP resource)**.** *A resource $\mathcal{S}$ is said to be a* remote state preparation resource *within ε with respect to converters $\mathcal{A}$ and $\mathcal{Q}$ if the following conditions hold:*

1. *Converters $\mathcal{A}$ and $\mathcal{Q}$ output a single message at the outer interface, where the output $[\rho]$ of $\mathcal{A}$ is classical and is either a density matrix or the null matrix, and the output ρ' of $\mathcal{Q}$ is a quantum state;*

2. *The following accuracy relation is satisfied:*

$$\mathop{\mathbb{E}}_{([\rho],\rho')\leftarrow\mathcal{A}\mathcal{S}\vdash\mathcal{Q}}[\mathrm{Tr}(\rho\rho')] \geq 1-\varepsilon \tag{6.2}$$

   where the probability is taken over the randomness of $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{Q}$

3. *For all the possible outputs $[\rho]$ of $([\rho],\rho') \leftarrow \mathcal{A}\mathcal{S} \vdash \mathcal{Q}$, if we define $E_0 = \rho$, $E_1 = I - \rho$, then the POVM $\{E_0, E_1\}$ must be efficiently implementable[6] by any distinguisher.*

In the remaining of the chapter, when we speak of an RSP resource S, this has to always be interpreted in a context where converters $\mathcal{A}$ and $\mathcal{Q}$ are fixed.
Additionally, to provide some intuition on the generality of this definition, the 2 converters would also allow us to define as an RSP: i) a resource in which the Client is sending the description of the state to the resource, where converter $\mathcal{A}$ will forward this description from its right interface to its left interface, or ii) a more complex and

[4] While we emphasized that the relevant RSP resources are the ones with high purity of the produced quantum states, our main result regarding the characterization of RSP resources hold also when the underlying quantum state is mixed (hence for any quantum state).

[5] This expression measuring the closeness between the states is also equal to the squared fidelity between ρ and ρ', when ρ is pure.

[6] We could also define a similar definition when it suffices that the POVM can to be approximated (for example because the distinguishers can only perform quantum circuits using a finite set of gates) and the results would be similar, up to this approximation, but for simplicity we will stick to this definition.

interesting example would be the classical client UBQC functionality which we will see in the next section on how it can be turned into an RSP resource.

Describable resources. We have defined that a resource qualifies as an RSP, if when both Client and Server follow the protocol, we know how to obtain a quantum state on the right interface and a classical description of a close state on the left interface. A security-related question is whether it is also possible to extract, possibly in an inefficient manner, from the right interface a classical description of a quantum state that is close to the state described by the output of Client. If there exists a converter $\mathcal{P}$ achieving this, we call the RSP resource *describable*. We formalize this as follows.

Definition 6.2.3 (Describable Resource)**.** *Let $\mathcal{S}$ be a resource and $\mathcal{A}$ a converter outputting a single classical message $[\rho]$ on its outer interface (either equal to a density matrix or the null matrix). Then we say that $(\mathcal{S},\mathcal{A})$ is* ε-describable *(or, equivalently, that $\mathcal{S}$ is describable within ε with respect to $\mathcal{A}$) if there exists a (possibly inefficient) converter $\mathcal{P}$ (outputting a single classical message $[\rho']$ on its outer interface representing a density matrix) such that:*

$$\mathop{\mathbb{E}}_{([\rho],[\rho'])\leftarrow\mathcal{A}\mathcal{S}\mathcal{P}}[\mathrm{Tr}(\rho\rho')] \geq 1-\varepsilon \tag{6.3}$$

(the expectation is taken over the randomness of $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{P}$).

Reproducible converters. To show our first main result about the characterization of RSP resources we will encounter a crucial decoding step. The core of this decoding element is the ability to convert the classical the classical interaction between Server and Client - which can be seen as an encoding of the quantum state - back into an explicit classical representation of the state prepared by the Server. More formally, this can be phrased in the following definition.

Definition 6.2.4 (Reproducible Converter)**.** *A converter π that outputs (on the right interface) a quantum state ρ is said to be reproducible if there exists a (possibly inefficient) converter $\tilde{\pi}$ such that:*

1. *The outer interface of $\tilde{\pi}$ outputs only a classical message $[\rho']$*
2. *The converter π is perfectly indistinguishable from $\tilde{\pi}$ against any unbounded distinguisher D, up to the conversion of the classical messages $[\rho']$ into a quantum state ρ'. More precisely, if we denote by $\mathcal{T}$ the converter that takes as input on*

its inner interface a classical description $[\rho']$ *of a quantum state and outputs that quantum state* ρ'*, we have (as shown in Figure 6.3):*

$$\mathcal{C}\pi \approx_0^D \mathcal{C}\tilde{\pi}\mathcal{T} \tag{6.4}$$

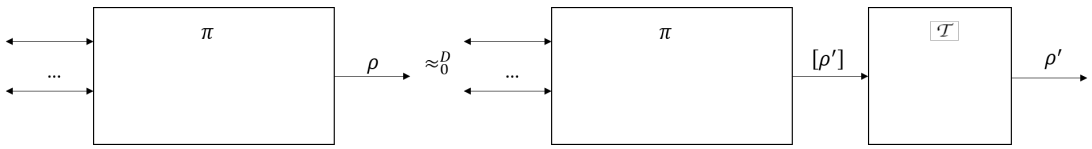


Figure 6.3: Reproducible converter π

Next, we will prove that whenever we have only classical communication, it is always possible to extract (in exponential time) the exact description of the state from the classical transcript and the *quantum instruments* (circuit) used to implement the actions of the converter. We recall from Definition 2.1.2 that a quantum instrument is a generalization of the CPTP map allowing for both a quantum and classical output.
In the proof of this result we consider that π interacts classically with the inner interface and then outputs a quantum state on the outer interface. In this way, we can decompose π, as depicted in Figure 6.4, using the notation:

$$\pi := (\pi_i)_i \tag{6.5}$$

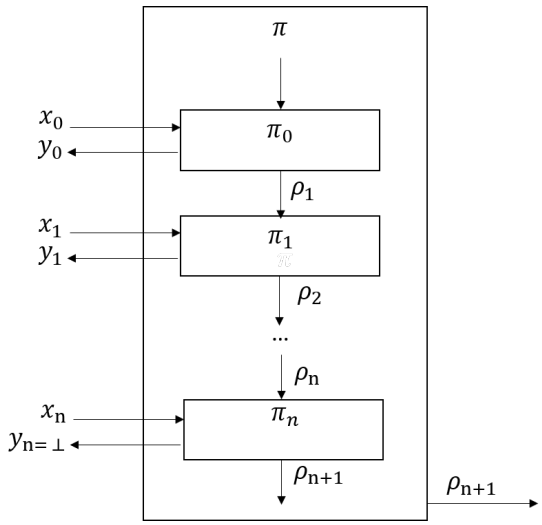


Figure 6.4: Representation of an interactive protocol π into a sequence of quantum instruments.

Each π_i is associated to a round of communication and we denote with $(y_i, \rho_{i+1}) \leftarrow \pi_i(x_i, \rho_i)$ the output of i-th round, where $x_i \in \{0,1\}^{l_i}$ is a classical input message received from the inner interface, ρ_i and ρ_{i+1} are the internal quantum state before and

respectively after round i and $y_i \in \{0,1\}^{l'_i} \cup \perp$ is a classical message sent by π to the inner interface, when $y_i \neq \perp$. Before the first round, we will set $\rho_0 = (1)$, which is the trivial density matrix of dimension 1. Moreover when $y_i = \perp$ this represents that we do not send any message anymore to the inner interface (the last round of communication) and instead we send ρ_{i+1} to the outer interface and stop the protocol. Note that if we want to let π send the first message instead of receiving it, we can simply set $x_0 = \perp$, and similarly if the last message is in fact sent instead of received, we can add one more round where we set $x_{n+1} = \perp$.
Now, we can finally show that a party producing a quantum state at the end of a protocol with exclusively classical communication, is reproducible.

Lemma 6.2.5. *Let $\pi = (\pi_i)_i$ be a converter such that:*

1. *π receives and sends only classical messages from the inner interfaces;*
2. *π outputs at the end a quantum state on the outer interface;*
3. *Each π_i is a quantum instrument;*

Then π is reproducible.

Proof. The intuition behind the proof is to argue that because the only interations of π are classical as seen in Figure 6.4 , the internal state of π can always be simulated (computed) in exponential time manually.

More precisely, for any i, as π_i is a quantum instrument (Definition 2.1.2) there exists a set of maps $\{\mathcal{E}_{y_i}\}$ such that:

- $\mathrm{Tr}\big(\sum_{y_i}(\rho)\big) = \mathrm{Tr}(\rho)$ for any quantum state ρ;
- If we denote by $\rho_{y_i} = \frac{\mathcal{E}_{y_i}(\rho)}{\mathrm{Tr}\big(\mathcal{E}_{y_i}(\rho)\big)}$, then we have:

$$\Pr[\pi(\rho) = (y_i, \rho_{y_i})] = \mathrm{Tr}(\mathcal{E}_{y_i}(\rho)) \text{ for any state } \rho \tag{6.6}$$

And because for every y_i, $\mathcal{E}_{y_i}$ is completely positive, there exists a finite set of matrices $\{B_k^{(i,y_i)}\}_k$, known as Kraus operators, such that for all ρ, we can decompose the action of $\mathcal{E}_{y_i}$ as:

$$\mathcal{E}_{y_i}(\rho) = \sum_k B_k^{(i,y_i)} \rho B_k^{(i,y_i)\dagger} \tag{6.7}$$

By choosing the state $\rho := |x_i\rangle\langle x_i| \otimes \rho_i$, from Eq.6.6, we obtain that with probability $p_{y_i} = \mathrm{Tr}(\mathcal{E}_{y_i}(|x_i\rangle\langle x_i| \otimes \rho_i))$ we have:

$$\pi_i(x_i, \rho_i) = (y_i, \mathcal{E}_{y_i}(|x_i\rangle \langle x_i| \otimes \rho_i)) \tag{6.8}$$

$$= (y_i, \underbrace{\sum_k B_k^{(i,y_i)}(|x_i\rangle \langle x_i| \otimes \rho_i) B_k^{(i,y_i)\dagger}}_{\rho_{i+1}}) \tag{6.9}$$

We remark that if we know $[\rho_i]$, the elements of the density matrix ρ_i, then we can simulate the output probability distribution of π, by determining for all y_i the probability p_{y_i} of outputting y_i and the corresponding state description $[\rho_{i+1}]$ (the coefficients of the density matrix ρ_{i+1}), by just doing the above classical computation. Therefore, to construct the converter $\tilde{\pi}$ we do the following:

- For every i, we construct $\tilde{\pi}_i$, which given an input $(x_i, [\rho_i])$ will output $(y_i, [\rho_{i+1}])$ with probability p_{y_i} using the formula in Eq.(6.9);
- We define $\tilde{\pi}$ as $\tilde{\pi} = (\tilde{\pi}_i)_i$, with $[\rho_0] = (1)$.

As a result, we trivially have $\mathcal{C}\pi \approx_0 \mathcal{C}\tilde{\pi}\mathcal{T}$ even against unbounded distinguishers, as $\tilde{\pi}$ is exactly simulating π, except that the representation of the quantum states in $\tilde{\pi}$ are matrices, while in π they are actual quantum states. Hence, by adding the converter $\mathcal{T}$ turning $[\rho_i]$ into ρ_i on the outer interface we obtain $\mathcal{C}\pi \approx_0 \mathcal{C}\tilde{\pi}\mathcal{T}$.

□

6.2.2 Classically-Realizable RSP are Describable

Now we are able to show our main result about RSP resources, which interestingly links a constructive computational notion (*composability*) with an information theoretic property (*describability*).

This directly implies the *impossibility result* regarding the existence of non-describable $\mathsf{RSP}_{\mathsf{CC}}$ composable protocols secure against bounded quantum polynomial-time distinguishers. While this no-go does not rule out all the possible RSP resources, it shows that the "useful" RSP resources are impossible. This is because the describability property is usually not a desirable property, as it implies that an unbounded adversary could learn the description of the state he received from an ideal resource. To illustrate the implication of this theorem, we will show in subsection 6.2.3 the impossibility of classical protocols implementing specific RSP resources and in subsubsection 6.2.4.1 we will see examples of "imperfect" resources escaping the impossibility result.

Theorem 6.2.6 (Classically-Realizable RSP are Describable)**.** *If an ideal resource $\mathcal{S}$ is both an ε_1-remote state preparation with respect to some $\mathcal{A}$ and $\mathcal{Q}$ and ε_2-classically-realizable (including against only polynomially bounded distinguishers), then it is $(\varepsilon_1+2\varepsilon_2)$-describable with respect to $\mathcal{A}$. In particular, if $\varepsilon_1=\mathsf{negl}(n)$ and $\varepsilon_2=\mathsf{negl}(n)$, then $\mathcal{S}$ is describable within a negligible error $\varepsilon_1+2\varepsilon_2=\mathsf{negl}(n)$.*

Proof. Let $\mathcal{S}$ be an ε_1-remote state preparation with respect to some converters $\mathcal{A}$ and $\mathcal{Q}$ which is also ε_2-classically-realizable. Then from Definition 6.2.2 and Definition 6.2.1, there exist π_A,π_B and simulator σ such that:

$$\mathop{\mathbb{E}}_{([\rho],\rho')\leftarrow\mathcal{A}\mathcal{S}\vdash\mathcal{Q}}[\mathrm{Tr}(\rho\rho')]\geq 1-\varepsilon_1 \tag{6.10}$$

$$\pi_A\mathcal{C}\pi_B\approx_{\varepsilon_2}\mathcal{S}\vdash \tag{6.11}$$

and

$$\pi_A\mathcal{C}\approx_{\varepsilon_2}\mathcal{S}\sigma \tag{6.12}$$

From Eq.(6.11), we also have:

$$\mathcal{A}\pi_A\mathcal{C}\pi_B\mathcal{Q}\approx_{\varepsilon_2}\mathcal{A}\mathcal{S}\vdash\mathcal{Q} \tag{6.13}$$

In other words we cannot distinguish between $\mathcal{A}\mathcal{S}\vdash\mathcal{Q}$ and $\mathcal{A}\pi_A\mathcal{C}\pi_B\mathcal{Q}$ with an advantage better than ε_2 (i.e. with probability better than $\frac{1}{2}(1+\varepsilon_2)$).
But consider the following efficient distinguisher:

1. Runs $([\rho],\rho')\leftarrow\mathcal{A}\mathcal{S}\vdash\mathcal{Q}$;
2. Then measures ρ' using the POVM $\{E_0,E_1\}$, where $E_0=\rho$ and $E_1=I-\rho$ (which is efficient to perform from the definition of remote state preparation);

Then, from Eq.(6.10), distinguisher will obtain outcome 0 (corresponding to E_0) with probability at least $1-\varepsilon_1$.
This means that by replacing $\mathcal{A}\mathcal{S}\vdash\mathcal{Q}$ with $\mathcal{A}\pi_A\mathcal{C}\pi_B\mathcal{Q}$, the probability to obtain outcome 0 (E_0) when $([\rho],\rho')\leftarrow\mathcal{A}\pi_A\mathcal{C}\pi_B\mathcal{Q}$, should also be close to $1-\varepsilon_1$. More precisely, we can show that we must have:

$$\mathop{\mathbb{E}}_{([\rho],\rho')\leftarrow\mathcal{A}\pi_A\mathcal{C}\pi_B\mathcal{Q}}[\mathrm{Tr}(\rho\rho')]\geq 1-\varepsilon_1-\varepsilon_2 \tag{6.14}$$

Assume by contradiction that the above probability is less than $1-\varepsilon_1-\varepsilon_2$.
Then we can construct a distinguisher $\mathcal{D}$ to distinguish between $\mathcal{A}\mathcal{S}\vdash\mathcal{Q}$ and $\mathcal{A}\pi_A\mathcal{C}\pi_B\mathcal{Q}$.
Essentially, $\mathcal{D}$ is defined as above: will simply measure the state ρ' using POVM

$\{E_0, E_1\}$. Then $\mathcal{D}$ will output the measurement outcome (0 for E_0 and 1 for E_1). Then, by denoting measurement outcome with m and the case $\mathcal{AS} \vdash Q$ with $a = 0$ and the case $\mathcal{A}\pi_A \mathcal{C}\pi_B Q$ with $a = 1$, the probability of $\mathcal{D}$ to distinguish $\mathcal{AS} \vdash Q$ from $\mathcal{A}\pi_A \mathcal{C}\pi_B Q$ is equal to:

$$\Pr[m = a] = \Pr[m = a \,|\, a = 0] \cdot \Pr[a = 0] + \Pr[m = a \,|\, a = 1] \cdot \Pr[a = 1] \tag{6.15}$$

$$= \frac{1}{2} \mathop{\mathbb{E}}_{([\rho],\rho') \leftarrow \mathcal{AS} \vdash Q} [\mathrm{Tr}(\rho\rho')] + \frac{1}{2} \mathop{\mathbb{E}}_{([\rho],\rho') \leftarrow \mathcal{A}\pi_A \mathcal{C}\pi_B Q} [\mathrm{Tr}((I - \rho)\rho')] \tag{6.16}$$

$$> \frac{1}{2} ((1 - \varepsilon_1) + 1 - (1 - \varepsilon_1 - \varepsilon_2)) \tag{6.17}$$

$$= \frac{1}{2}(1 + \varepsilon_2) \tag{6.18}$$

Therefore, distinguisher $\mathcal{D}$ has an advantage greater than ε_2, which is in contradiction with Equation 6.13.

Now, from Eq.(6.12), we also have:

$$\mathcal{A}\pi_A \mathcal{C}\pi_B Q \approx_{\varepsilon_2} \mathcal{AS}\sigma\pi_B Q \tag{6.19}$$

Similarly, we will construct a distinguisher $\mathcal{D}'$ between $\mathcal{A}\pi_A \mathcal{C}\pi_B Q$ and $\mathcal{AS}\sigma\pi_B Q$: gets $([\rho], \rho')$ and measures ρ' with POVM E_0, E_1 and outputs the measurement outcome. As before this would imply that:

$$\mathop{\mathbb{E}}_{([\rho],\rho') \leftarrow \mathcal{AS}\sigma\pi_B Q} [\mathrm{Tr}(\rho\rho')] \geq \mathop{\mathbb{E}}_{([\rho],\rho') \leftarrow \mathcal{A}\pi_A \mathcal{C}\pi_B Q} [\mathrm{Tr}(\rho\rho')] - \varepsilon_2 \geq 1 - \varepsilon_1 - 2\varepsilon_2 \tag{6.20}$$

We will now use the converter $\pi_B Q$ to construct a $\mathcal{B}$ that can describe the state given by the ideal resource $\mathcal{S}$. Because $\pi_B Q$ interacts only classically with the inner interface (with the simulator σ) and outputs a quantum state on the outer interface, then using Lemma 6.2.5, this implies that $\pi_B Q$ is reproducible. This means that there exists an inefficient converter $\mathcal{B}$ such that:

$$\mathcal{C}\pi_B Q \approx_0 \mathcal{CBT} \tag{6.21}$$

Therefore, we also have in particular $\mathcal{AS}\sigma\mathcal{C}\pi_B Q \approx_0 \mathcal{AS}\sigma\mathcal{CBT}$, and because $\mathcal{C}$ is a neutral resource we can remove $\mathcal{C}$, which implies:

$$\mathop{\mathbb{E}}_{([\rho],\rho') \leftarrow \mathcal{AS}\sigma\mathcal{BT}} [\mathrm{Tr}(\rho\rho')] \geq 1 - \varepsilon_1 - 2\varepsilon_2 \tag{6.22}$$

But because $\mathcal{T}$ simply converts the classical description $[\rho']$ into ρ', we also have:

$$\mathop{\mathbb{E}}_{([\rho],[\rho']) \leftarrow \mathcal{AS}\sigma\mathcal{B}} [\mathrm{Tr}(\rho\rho')] \geq 1 - \varepsilon_1 - 2\varepsilon_2 \tag{6.23}$$

Finally, by defining the converter $\mathcal{P}$ as $\mathcal{P} := \sigma\mathcal{B}$, we obtain that $(\mathcal{S}, \mathcal{A})$ is $(\varepsilon_1 + 2\varepsilon_2)$-describable (where $\mathcal{P}$ is the converter describing the quantum state), which concludes the proof.

□

6.2.3 RSP Resources Impossible to Realize Classically

In the previous section we show that if an RSP functionality is classically-realizable (secure against polynomial quantum distinguishers), then this resource is describable by an unbounded adversary having access to the right interface of that resource.

This main result directly implies that as soon as there exists *no unbounded* adversary that, given access to the right interface, can find the classical description given on the left interface, then the RSP resource is *impossible* to classically realize (against *bounded* BQP distinguishers). Very importantly, this no-go result shows that the *only* type of RSP resources that can be classically realized are the ones that *leak* on the right interface enough information to allow an (possibly unbounded) adversary to determine the classical description given on the left interface. From a security point of view, this property is highly non-desirable, as the resource must leak the *secret description* of the state at least in *some representation*.

We now present some examples of RSP resource impossible to classically realize.

Definition 6.2.7 (Ideal Resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$)**.** *$\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is the verifiable* RSP *resource (*RSP *which does not allow any deviation from the server), that receives no input, that internally picks a random $\theta \leftarrow \mathbb{Z}_{\frac{\pi}{2}}$, and that sends θ on the left interface, and $|+_\theta\rangle$ on the right interface as shown in Figure 6.5.*

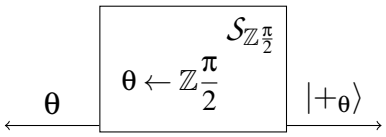


Figure 6.5: Ideal resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$

Lemma 6.2.8. *There exists a universal constant $\eta > 0$, such that for all $0 \leq \varepsilon < \eta$ the resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is not ε-classically-realizable.*

Proof. The impossibility proof at its core will be a consequence of quantum no-cloning.

We begin with defining $\mathcal{A}(\theta) := [|+_\theta\rangle\langle+_\theta|]$ ($\mathcal{A}$ just converts θ into its classical density matrix representation) and Q the trivial converter that just forwards any message from its inner to its outer interface. Then we have that $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is a 0-remote state preparation resource with respect to $\mathcal{A}$ and Q as:

$$\mathop{\mathbb{E}}_{([\rho],\rho')\leftarrow\mathcal{A}\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}\vdash Q}[]\operatorname{Tr}\left(\rho\rho'\right)] = \frac{1}{4}\sum_{\theta\in\mathbb{Z}_{\frac{\pi}{2}}}\operatorname{Tr}(|+_\theta\rangle\langle+_\theta|\,|+_\theta\rangle\langle+_\theta|) = 1 \geq 1-0 \tag{6.24}$$

Then, we will show that there exists a constant $\eta > 0$ such that for all $\delta < \eta$, $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is not δ-describable with respect to $\mathcal{A}$.

We first prove that $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is not 0-describable with respect to $\mathcal{A}$. We prove this by contradiction, hence assuming that there exists a converter $\mathcal{P}$ such that:

$$\mathop{\mathbb{E}}_{([\rho],[\rho'])\leftarrow\mathcal{A}\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}\mathsf{P}}[\operatorname{Tr}\left(\rho\rho'\right)] = 1 \tag{6.25}$$

But then, because $\rho = |+_\theta\rangle\langle+_\theta|$ is a pure state, $\operatorname{Tr}(\rho\rho')$ corresponds to the fidelity of ρ and ρ', so $\operatorname{Tr}(\rho\rho') = 1 \Leftrightarrow \rho = \rho'$. However this is impossible because P just has a quantum state ρ as input, and if he can completely describe this quantum state then he can actually clone perfectly the input state with probability 1. But because the different possible values of ρ are not orthogonal, this is impossible due to the no-cloning theorem.

This tells us that the resource $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ cannot be 0-classically-realizable. Specifically, if we assume by contradiction that $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is 0-classically-realizable and since $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is a 0-remote state preparation, then from Theorem 6.2.6, we would have that $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is 0-describable, reaching a contradiction.

What we want to show is something stronger, namely that $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is not $\mathsf{negl}(n)$-classically realizable.

From the optimality of quantum no-cloning [SIGA05, FWJ+14] we also know we cannot produce two copies of ρ with a fidelity arbitrary close to 1.

Therefore, there exists a constant $\eta > 0$, such that:

$$\mathop{\mathbb{E}}_{([\rho],[\rho'])\leftarrow\mathcal{A}\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}\mathsf{P}}[\operatorname{Tr}\left(\rho\rho'\right)] < 1-\eta \tag{6.26}$$

Now, by contradiction, we assume that $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is ε-classically-realizable. Because $\lim_{n\to\infty}\varepsilon(n) = 0$, there exists $N \in \mathbb{N}$ such that $\varepsilon(N) < \eta/2$.

Therefore, using Theorem 6.2.6, $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ is $0+2\varepsilon(N)$-describable with respect to $\mathcal{A}$. From the definition of describability this is equivalent to:

$$\mathop{\mathbb{E}}_{([\rho],[\rho'])\leftarrow\mathcal{A}\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}\mathsf{P}}[\operatorname{Tr}\left(\rho\rho'\right)] \geq 1-2\varepsilon(N) > 1-\eta \tag{6.27}$$

which contradicts Eq.(6.26) and completes our proof.

□

Next, we will describe the resource RSP_V, a variant of $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ introduced in [GV19]. This resource differs from $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ in the following aspects: the adversary can make the resource abort, the set of output states is larger and the client can choose the basis of the output state.
As for $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$, we will show that classically-realizable RSP_V cannot be achieved. Before presenting the details of the no-go result, we formalize the ideal resource RSP_V.

Definition 6.2.9 (Ideal Resource RSP_V)**.** *The ideal verifiable remote state preparation resource* RSP_V*, takes an input* $W \in \{X,Z\}$ *on the left interface, and no input on the right interface. The right interface has a filtered functionality that corresponds to a bit* $c \in \{0,1\}$*. When* $c = 1$*,* RSP_V *outputs error message* ERR *on both the interfaces, otherwise:*

1. *If* $W = Z$*, resource picks a random bit* b *and outputs* b *to the left interface and the state* $|b\rangle\langle b|$ *to the right interface;*

2. *If* $W = X$*, resource picks a random angle* $\theta \in \mathbb{Z}_{\frac{\pi}{4}}$ *and outputs* θ *to the left interface and the quantum state* $|+_\theta\rangle\langle+_\theta|$ *to the right interface.*

Corollary 6.2.10. *There exists a universal constant* $\eta > 0$*, such that for all* $0 \leq \varepsilon < \eta$ *the resource* RSP_V *is not* ε*-classically-realizable.*

Proof. To show this we follow exactly the steps of the impossibility proof for $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$. The main difference is that we need to address the abort case occurring when $c = 1$. To do this, we will change the converter $\mathcal{A}$ as follows: $\mathcal{A}$ will pick $W = X$ and will output to the outer interface either the classical density matrix corresponding to $\rho = |+_\theta\rangle\langle+_\theta|$, when $c = 0$ or the null matrix $\rho = 0$ when $c = 1$ (ERR). Q remains as in the case of $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$, the trivial converter. Now, it is easy to see that RSP_V is a 0-remote state preparation resource with respect to converters $\mathcal{A}$ and Q. Moreover, RSP_V cannot be ε-describable for arbitrary small ε, as when $c = 1$, we have $\rho = 0$ thus $Tr(\rho\rho')$ is 0. Hence, from a converter $\mathcal{P}$ that can also input $c = 1$, we can always increase the quantity $Tr(\rho\rho')$, by considering a new converter $\mathcal{P}'$ setting $c = 0$. Then, we are essentially back to the setting of $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$, where we have a the set of states $|+_\theta\rangle\langle+_\theta|$ that are impossible to be cloned with arbitrary small fidelity, which concludes the proof. □

Remark 6.2.11. *Note that our impossibility result for classical realization of* $\mathsf{RSP}_{\mathsf{V}}$ *does not contradict the result of [GV19]. Specifically, their security analysis requires an assumption of a "measurement buffer"*[7] *resource in addition to the classical channel in order to construct* $\mathsf{RSP}_{\mathsf{V}}$*. Our result shows that the measurement buffer resource is a strictly non-classical assumption.*

6.2.4 Characterization of RSP resources

The main result in Theorem 6.2.6 rules out all resources that are impossible to be *describable* with unbounded power such as $\mathcal{S}_{\mathbb{Z}_{\frac{\pi}{2}}}$ or $\mathsf{RSP}_{\mathsf{V}}$. More importantly, it tells us the only type of classically-realizable RSP resources must be describable and hence would be the ones leaking the full classical description of the output quantum state to an unbounded adversary, which we will refer to as being *fully-leaky* RSP.
Fully-leaky RSP resources can be separated into two categories:

1. If the RSP is describable in quantum polynomial time, then the adversary can learn the secret in polynomial time. This is obviously not an interesting case as the security of protocols such as UBQC cannot be preserved if such a resource is employed to prepare the quantum states.

2. If the RSP are only describable using unbounded power, then these *fully-leaky* RSP resources are not trivially insecure, but their composable security remains unclear. Indeed, it defeats the purpose of aiming at a proper ideal resource where the provided security should be clear "by definition" and it becomes hard to quantify how the additional leakage could be used when composed with other protocols.

To complete the picture of RSP resources, we will next show an example of a fully-leaky RSP denoted by $\mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}}$, describable only with unbounded power, together with a protocol that realizes it. This resource is inspired from the Malicious 4-states QFactory presented in chapter 5 and this protocol will be precisely the example realizing it. As a final remark, we emphasize that this category of leaky RSP resources is not desirable: the resources are non-standard and it looks hard to write a modular protocol with this resource as an assumed resource. Additionally, the resource is very specific and mimics its implementation.

[7] This resource forces distinguisher to give the state he is supposed to measure to the simulator, allowing the simulator to change the state given by the distinguisher with the state sent by the ideal resource, without letting the distinguisher know.

6.2.4.1 Fully-Leaky RSP

Definition 6.2.12 (Ideal Resource $\mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}}$)**.** *Let $\mathcal{F} = (Gen, Enc, Dec)$ be a public-key encryption scheme. Then, $\mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}}$ is defined as depicted in Figure 6.6. Specifically, B_1 represents the basis bit of the output state, and is guaranteed to be random even when the server on the right interface is malicious. B_2 represents the value bit of the output state when encoded in the basis B_1, and in the malicious case ($c = 1$) it can be chosen by the right interface, expressed through the deviation function D (as seen in Figure 6.6)*[8]*. The resource sends to the left interface the classical description (B_1, B_2) and, in an honest run, sends to the right interface the quantum state $|\psi\rangle := H^{B_1} X^{B_2} |0\rangle$.*

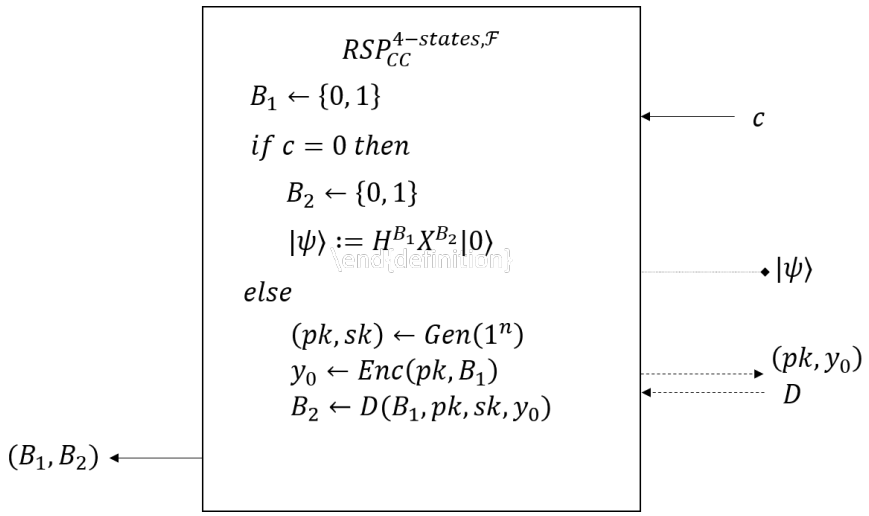


Figure 6.6: Ideal resource $\mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}}$ which prepares a BB84 state. The state $|\psi\rangle$ is sent to the right interface only in the honest case ($c = 0$) and the dashed communication is exchanged only in the dishonest case ($c = 1$).

Lemma 6.2.13. *The Malicious 4-states QFactory protocol (Protocol 3) securely constructs $\mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}}$ from a classical channel.*

Proof. We define $\mathcal{F}$ based on the family of functions required for the construction of Malicious 4-states QFactory, as described in subsection 5.3.2. Concretely, we have:

1. $Gen(1^n) \rightarrow (K, t_K)$, where K is the function description and t_K is the trapdoor information;

[8] Note that for $\mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}}$ the right interface (Server) can have in a malicious scenario full control over B_2, but in the Malicious 4-states QFactory Protocol it is not clear what an adversary can do concerning B_2.

2. $Enc_K(B_1) \to y_0$, where $y_0 = Ks_0 + e_0 + B_1 \cdot v$, where s_0, e_0 and v are chosen as in subsection 5.3.2;

3. $Dec_{t_K}(y_0) \to B_1$ - using t_K we can efficiently obtain B_1 from y_0

From the correctness of the QFactory protocol (π_A, π_B), shown in Theorem 5.2.2, we have:

$$\pi_A \mathcal{C} \pi_B \approx_\varepsilon \mathsf{RSP}_{\mathsf{CC}}^{4-\mathsf{states},\mathcal{F}} \vdash \tag{6.28}$$

for some negligible ε.

We now need to show that there exists a simulator σ such that:

$$\pi_A \mathcal{C} \approx_{\varepsilon'} \mathsf{RSP}_{\mathsf{CC}}^{4\text{-states},\mathcal{F}} \sigma \tag{6.29}$$

Simulator σ is constructed as follows: sends $c = 1$ to the ideal resource, then forwards the message (K, y_0) received from the resource to its outer interface and when receiving the measurement outcomes (y, b) from the server (as in the real protocol) it sets the deviation function D to be the same function as Client uses in π_A to obtain B_1. As a result, we trivially have: $\pi_A \mathcal{C} \approx_0 \mathsf{RSP}_{\mathsf{CC}}^{4\text{-states},\mathcal{F}} \sigma$, which concludes the proof. □

6.3 Impossibility of Composable Classical-Client UBQC

In the previous section we showed that it is impossible to have a useful composable $\mathsf{RSP}_{\mathsf{CC}}$ protocol. However, a weaker RSP protocol could still be used internally in other protocols, with the aim that the overall protocol to be composably secure. To this end, we analyze the composable security of a well-established delegated quantum computing protocol, universal blind quantum computation (UBQC), proposed in [BFK09]. The UBQC protocol allows a quantum client, Alice, to delegate an arbitrary quantum computation to a (universal) quantum server Bob, in such a way that her input, the quantum computation and the output of the computation are information-theoretically hidden from Bob. The protocol requires Alice to be able to prepare single qubits of the form $|+_\theta\rangle$, where $\theta \in \mathbb{Z}\frac{\pi}{4}$ and send these states to Bob at the beginning of the protocol, the rest of the communication between the two parties being classical.

We define the family of protocols $\mathsf{RSP}_{\mathsf{CC}}^{8-\mathsf{states}}$ as the RSP protocols that classically delegate the preparation of an output state $|+_\theta\rangle$, where $\theta \in \mathbb{Z}\frac{\pi}{4}$. That is, without loss of generality, we assume a pair of converters P_A, P_B such that the resource $R := P_A \mathcal{C} P_B$ has the behavior of the prototype RSP resource except with negligible probability. Put differently, we assume we have an (except with negligible error) *correct* RSP protocol,

but we make *no assumption about the security* of this protocol. Therefore, we can directly replace the quantum interaction with the $\mathsf{RSP}_{\mathsf{CC}}^{8-\mathsf{states}}$ as indicated in the first step of Protocol 7 presented below, and obtain a new protocol for the delegation of quantum computations between a quantum server and an entirely classical client.

Protocol 7 UBQC with $\mathsf{RSP}_{\mathsf{CC}}^{8-\mathsf{states}}$ ([BFK09])

- **Client's classical input:** An n-qubit unitary U that is represented using the set of angles $\{\phi\}_{i,j}$ of a one-way quantum computation over a brickwork state/cluster state [MDF17], of size $n \times m$, along with the dependencies X and Z obtained via flow construction [DK06].
- **Client's classical output:** The measurement outcome $\bar{s}$ corresponding to the n-qubit quantum state, where $\bar{s} = \langle 0| U |0\rangle$.

1. Client and Server runs $n \times m$ different instances of $\mathsf{RSP}_{\mathsf{CC}}^{8-\mathsf{states}}$ (in parallel) to obtain $\theta_{i,j}$ on client's side and $|+_{\theta_{i,j}}\rangle$ on server's side, where $\theta_{i,j} \leftarrow \mathbb{Z}\frac{\pi}{4}$, $i \in \{1,\cdots,n\}$, $j \in \{1,\cdots,m\}$
2. Server entangles all the qubits, $n\cdot(m-1)$ received from $\mathsf{RSP}_{\mathsf{CC}}^{8-\mathsf{states}}$, by applying controlled-Z gates between them in order to create a graph state $\mathcal{G}_{n\times m}$
3. For $j \in [1,m]$ and $i \in [1,n]$
   (a) Client computes $\delta_{i,j} = \phi'_{i,j} + \theta_{i,j} + r_{i,j}\pi$, $r_{i,j} \leftarrow \{0,1\}$, where $\phi'_{i,j} = (-1)^{s^X_{i,j}}\phi_{i,j} + s^Z_{i,j}\pi$ and $s^X_{i,j}$ and $s^Z_{i,j}$ are computed using the previous measurement outcomes and the X and Z dependency sets. Client then sends the measurement angle $\delta_{i,j}$ to the Server.
   (b) Server measures the qubit $|+_{\theta_{i,j}}\rangle$ in the basis $\{\left|+_{\delta_{i,j}}\right\rangle, \left|-_{\delta_{i,j}}\right\rangle\}$ and obtains a measurement outcome $s_{i,j} \in \{0,1\}$. Server sends the measurement result to the client.
   (c) Client computes $\bar{s}_{i,j} = s_{i,j} \oplus r_{i,j}$.
4. The measurement outcome corresponding to the last layer of the graph state ($j = m$) is the outcome of the computation.

Note that Protocol 7 is based on measurement-based model of quantum computing (MBQC). This model is known to be equivalent to the quantum circuit (up to polynomial overhead in resources) and does not require one to perform quantum gates on their side to realize arbitrary quantum computation. Instead, the computation is performed by an (adaptive) sequence of single-qubit projective measurements that steer the information flow across a highly entangled resource state. Intuitively, UBQC can be seen as

a distributed MBQC where the measurements are performed by the server whereas the classical update of measurement bases is performed by the client. Since the projective measurements in quantum physics, in general, are probabilistic in nature and therefore, the client needs to update the measurement bases (and classically inform the server about the update) based on the outcomes of the earlier measurements to ensure the correctness of the computation. Roughly speaking, this information flow is captured by the X and Z dependencies. For more details, we refer the reader to [RB01, Nie06].

In the remaining of the section we will show that the Universal Blind Quantum Computing protocol, which is proven to be secure in the Constructive Cryptography framework [DFPR14], cannot be composably secure, for the same ideal resource, when the quantum interaction is replaced with a $\mathsf{RSP}_{\mathsf{CC}}$ protocol (this class of resulting protocols will be denoted as $\mathsf{UBQC}_{\mathsf{CC}}$).

6.3.1 Impossibility of Composable $\mathsf{UBQC}_{\mathsf{CC}}$ on 1 Qubit

In order to show that there can exist no composable $\mathsf{UBQC}_{\mathsf{CC}}$ protocol, we will first show the impossibility of a simpler setting, when the underlying quantum computation (in UBQC) is described by a single measurement angle. The corresponding resource that performs blind quantum computations on one qubit will be denoted by $\mathcal{S}_{UBQC1}$ and is defined as follows:

Definition 6.3.1 (Ideal resource of single-qubit UBQC)**.** *The ideal resource $\mathcal{S}_{UBQC1}$, depicted in Figure 6.7, achieves blind quantum computation, where the computation is specified by a single input angle ϕ. The input (ξ,ρ) is filtered when $c=0$. ξ represents any deviation (specified using the classical description of a CPTP map) outputting a bit, and which can depend on the computation angle ϕ and on an arbitrary state ρ.*

Theorem 6.3.2 (No-go composable classical-client single-qubit UBQC)**.** *Consider (P_A,P_B) be a protocol interacting only using a classical channel $\mathcal{C}$, such that the output of the protocol is P_A receiving classical output $\theta\in\mathbb{Z}_{\frac{\pi}{4}}$ and P_B receiving quantum output ρ_B (denoted as $(\theta,\rho_B)\leftarrow(P_A\mathcal{C})P_B$ and such that the trace distance between ρ_B and $|+_\theta\rangle\langle+_\theta|$ is negligible with overwhelming probability. Then if we define π_A and π_B as the UBQC protocol on one qubit that makes use of (P_A,P_B) as a sub-protocol in order to replace the quantum channel (as depicted in Figure 6.8), then the protocol (π_A,π_B) cannot be composable, i.e. there exists no simulator σ such that:*

$$\begin{aligned}&\pi_A\mathcal{C}\pi_B \approx_\varepsilon \mathcal{S}_{UBQC1} \vdash^{c=0}\\ &\pi_A\mathcal{C} \approx_\varepsilon \mathcal{S}_{UBQC1}\sigma \qquad\qquad \text{, where } \varepsilon=\mathsf{negl}(n)\end{aligned} \tag{6.30}$$

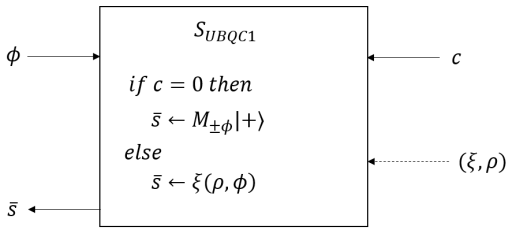


Figure 6.7: Ideal resource $\mathcal{S}_{UBQC1}$ for UBQC with computation described by one angle ϕ. In the case of honest server, the output $\bar{s} \in \{0,1\}$ is computed by measuring the qubits $|+\rangle$ in the $\{|+_\phi\rangle, |-_\phi\rangle\}$ basis. On the other hand, if $c = 1$, any malicious behaviour of server can be captured by (ξ, ρ), i.e. the output $\bar{s}$ is computed by applying the CPTP map ξ on the input ϕ and on another auxiliary state ρ chosen by the server.

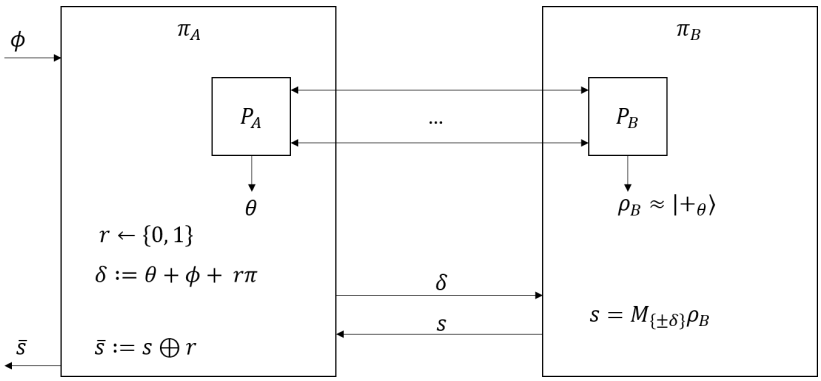


Figure 6.8: UBQC with $\mathsf{RSP}_{\mathsf{CC}}^{8-\mathsf{states}}$ protocol (P_A, P_B) on one qubit computation, when both Alice and Bob follow the protocol honestly (See Protocol 7). $M_{\pm\delta}$ represents a measurement in the basis $\{|+_\delta\rangle, |-_\delta\rangle\}$.

Sketch of Proof. In order to show this no-go result, we will proceed with a proof by contradiction. Let us assume there exists a protocol (P_A, P_B) and a simulator σ satisfying the above conditions. Then, for the same resource $\mathcal{S}_{UBQC1}$ we will consider a different protocol $\pi' = (\pi'_A, \pi'_B)$ that realizes it, but using a different filter [9] $\vdash^\sigma$ and a different simulator σ':

$$\pi'_A C \pi'_B \approx_\varepsilon \mathcal{S}_{UBQC1} \vdash^\sigma \tag{6.31}$$

$$\pi'_A C \approx_\varepsilon \mathcal{S}_{UBQC1} \sigma' \tag{6.32}$$

[9] Note that we could include this new filter $\vdash^\sigma$ inside the $\mathcal{S}_{UBQC1}$ resource and then have the standard filter $\vdash^{c=0}$, but for simplicity we will just use a different filter. Specifically, we could have defined a functionality $\mathcal{S}'_{UBQC1}$ that receives as input a bit c and if $c = 0$, then $\mathcal{S}'_{UBQC1}$ behaves as the resource $\mathcal{S}_{UBQC1} \vdash^\sigma$ and if $c = 1$ then it behaves as $\mathcal{S}_{UBQC1}$.

The new filter $\vdash^{\sigma}$ will depend on the simulator σ required for the soundness of (π_A, π_B), as seen in Eq.(6.30).

Then our full proof will follow the next steps:

1. We first prove in Lemma 6.3.4 that $\mathcal{S}_{UBQC1}$ is also ε-classically-realizable by (π'_A, π'_B) using the filter $\vdash^{\sigma}$.

2. We then show in Lemma 6.3.5 that the resource $\mathcal{S}_{UBQC1}$ is an RSP within $\mathsf{negl}(n)$ with respect to some well chosen converters $\mathcal{A}$ and Q (as depicted in Figure 6.10) and the filter $\vdash^{\sigma}$.

3. Then we use our main result about the characterization of RSP (Theorem 6.2.6) to deduce that $\mathcal{S}_{UBQC1}$ is describable within $\mathsf{negl}(n)$ with respect to converter $\mathcal{A}$ (Corollary 6.3.6).

4. Finally, we prove that if $\mathcal{S}_{UBQC1}$ is describable, then we can achieve *superluminal signaling*, concluding our contradiction proof (Lemma 6.3.8).

□

Now, let us first define the above mentioned protocol π' and the corresponding filter $\vdash^{\sigma}$, which we will show they classically-realize the $\mathcal{S}_{UBQC1}$ resource.

Definition 6.3.3 (π')**.** *Consider the protocol $\pi' = (\pi'_A, \pi'_B)$ defined in the following way (as depicted in Figure 6.9):*

- $\pi'_A = \pi_A$
- π'_B*:*
    1. *Runs P_B and obtains a state $\rho_B \approx |+_\theta\rangle$;*
    2. *Computes the quantum state $\tilde{\rho} := R_Z(-\delta)\rho_B$, using the angle δ received from π'_A;*
    3. *Outputs $s := 0$ on its inner interface (to π'_A) and the state $\tilde{\rho}$ on its outer interface.*

Then we define the corresponding filter as $\vdash^{\sigma} := \sigma\pi'_B$, where σ is the simulator corresponding to π (Eq.(6.30)).

Lemma 6.3.4. *If $\mathcal{S}_{UBQC1}$ is ε-classically-realizable by (π_A, π_B) with the filter $\vdash^{c=0}$ then $\mathcal{S}_{UBQC1}$ is ε-classically-realizable by (π'_A, π'_B) with the filter $\vdash^{\sigma}$.*

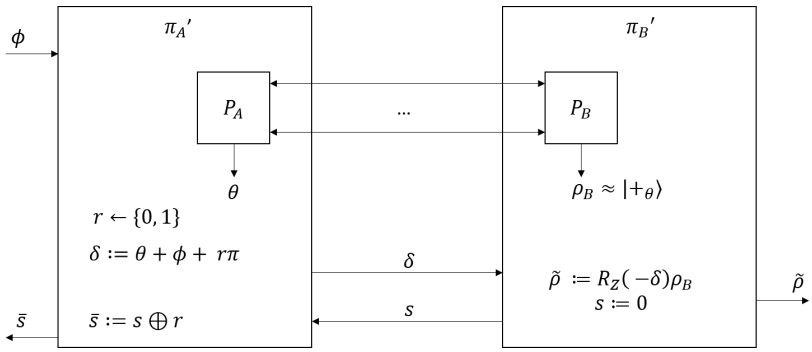


Figure 6.9: Protocol $\pi' = (\pi'_A, \pi'_B)$

Proof. If $\mathcal{S}_{UBQC1}$ is ε-classically-realizable with $\vdash^{c=0}$ by (π_A, π_B), then as seen in Theorem 6.3.2, we have:

$$\pi_A C \pi_B \approx_\varepsilon \mathcal{S}_{UBQC1} \vdash^{c=0} \tag{6.33}$$

$$\pi_A C \approx_\varepsilon \mathcal{S}_{UBQC1} \sigma \tag{6.34}$$

Now, we can show that $\mathcal{S}_{UBQC1}$ is ε-classically-realizable by (π'_A, π'_B), illustrated in Figure 6.9, together with $\vdash^\sigma$, namely that there exists a simulator σ' such that the following 2 conditions are satisfied:

$$\begin{aligned} \pi'_A C \pi'_B &\approx_\varepsilon \mathcal{S}_{UBQC1} \vdash^\sigma \\ \pi'_A C &\approx_\varepsilon \mathcal{S}_{UBQC1} \sigma' \end{aligned} \tag{6.35}$$

For the correctness condition by using the definitions of π'_A and $\vdash^\sigma$ and Eq.(6.34), we have:

$$\begin{aligned} \pi'_A C \pi'_B = \pi_A C \pi'_B &= (\pi_A C)\pi'_B \\ &\approx_\varepsilon (\mathcal{S}_{UBQC1}\sigma)\pi'_B = \mathcal{S}_{UBQC1}(\sigma\pi'_B) \\ &= \mathcal{S}_{UBQC1} \vdash^\sigma \end{aligned} \tag{6.36}$$

For the security condition, we will define the simulator as $\sigma' := \sigma$. Then, from Eq.(6.34) we obtain:

$$\pi'_A C = \pi_A C \approx_\varepsilon \mathcal{S}_{UBQC1}\sigma = \mathcal{S}_{UBQC1}\sigma' \tag{6.37}$$

which concludes our proof.

□

Lemma 6.3.5. *If $\mathcal{S}_{UBQC1}$ is $\mathsf{negl}(n)$-classically-realizable with $\vdash^{c=0}$, then $\mathcal{S}_{UBQC1}$ is an $\mathsf{negl}(n)$-remote state preparation resource with respect to converters $\mathcal{A}$ and Q defined in Figure 6.10, and the filter $\vdash^\sigma$.*

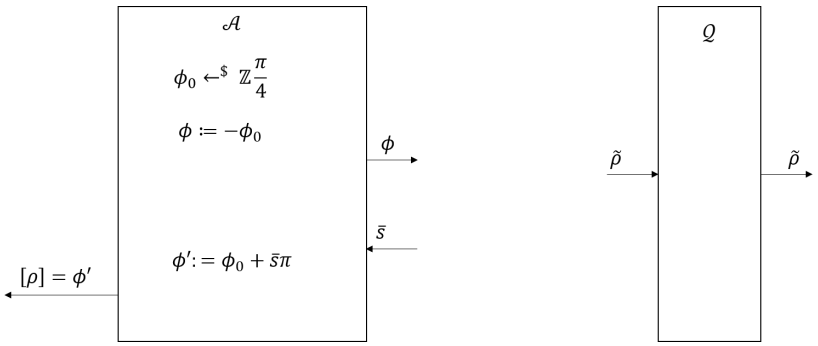


Figure 6.10: Converters $\mathcal{A}$ and Q

Proof. Given the converters $\mathcal{A}$ and Q defined in Figure 6.10, to show that $\mathcal{S}_{UBQC1}$ is a remote state preparation resource we need to prove that:

$$\mathop{\mathbb{E}}_{([\rho],\tilde{\rho})\leftarrow\mathcal{A}\mathcal{S}_{UBQC1}\vdash^{\sigma}Q}[\mathrm{Tr}(\rho\tilde{\rho})] \geq 1 - \mathsf{negl}(n) \tag{6.38}$$

Firstly, using Lemma 6.3.4, we also have:

$$\mathcal{A}\pi'_A\mathcal{C}\pi'_BQ \approx_{\mathsf{negl}(n)} \mathcal{A}\mathcal{S}_{UBQC1} \vdash^{\sigma} Q \tag{6.39}$$

Now let us examine the real world $\mathcal{A}\pi'_A\mathcal{C}\pi'_BQ$. Using the description of the protocol and of the 2 converters we have: $\bar{s} = 0 \oplus r = r$ and therefore the classical description output of $\mathcal{A}$ is:

$$\phi' = \phi_0 + \bar{s}\pi = -\phi + r\pi$$

And the corresponding state is $\rho = \left|+_{\phi'}\right\rangle\left\langle+_{\phi'}\right| = \left|+_{-\phi+r\pi}\right\rangle\left\langle+_{-\phi+r\pi}\right|$.
Then we analyze the output of Q on the rightmost interface. From the correctness of the protocol (P_A, P_B), we have that the trace distance between ρ_B and $|+_\theta\rangle\langle+_\theta|$ is negligible and as trace distance is preserved under unitary transformations, we also have that: $\tilde{\rho} = R_Z(-\delta)\rho_B R_Z(-\delta)^+$ is negligibly close in trace distance to the state:

$$\begin{aligned}
&R_Z(-\delta)\left|+_\theta\right\rangle\left\langle+_\theta\right|R_Z(-\delta)^+ = \left|+_{\theta-\delta}\right\rangle\left\langle+_{\theta-\delta}\right| \\
&= \left|+_{-\phi-r\pi}\right\rangle\left\langle+_{-\phi-r\pi}\right| \text{ (from the definition of } \delta) \\
&= \left|+_{-\phi+r\pi}\right\rangle\left\langle+_{-\phi+r\pi}\right| = \left|+_{\phi'}\right\rangle\left\langle+_{\phi'}\right| = \rho
\end{aligned}$$

As a result, we have:

$$\mathop{\mathbb{E}}_{([\rho],\tilde{\rho})\leftarrow\mathcal{A}\pi'_A\mathcal{C}\pi'_BQ}[\mathrm{Tr}(\rho\tilde{\rho})] \geq 1 - \mathsf{negl}(n) \tag{6.40}$$

Now, consider that:

$$\mathop{\mathbb{E}}_{([\rho],\tilde{\rho})\leftarrow\mathcal{A}\mathcal{S}_{UBQC1}\vdash^{\sigma}Q}[\mathrm{Tr}(\rho\tilde{\rho})] = 1 - \varepsilon \tag{6.41}$$

and we will need to show that $\varepsilon = \mathsf{negl}(n)$.

Using a similar argument to the one given in Theorem 6.2.6, we will define the following distinguisher $\mathcal{D}$ to distinguish between the ideal $\mathcal{AS}_{UBQC1} \vdash^{\sigma} Q$ and the real world $\mathcal{A}\pi'_A C\pi'_B Q$. More specifically, $\mathcal{D}$ will obtain $([\rho, \tilde{\rho}])$ (from the real or ideal world) and he will measure the state $\tilde{\rho}$ using the POVM $\{E_0 = [\rho], E_1 = I - [\rho]\}$. Then $\mathcal{D}$ will output the measurement outcome $1-m$ ($m = 0$ - corresponding to E_0 or $m = 1$ - corresponding to E_1). Then the probability p of $\mathcal{D}$ to distinguish between $\mathcal{AS}_{UBQC1} \vdash^{\sigma} Q$ $(a = 0)$ and $\mathcal{A}\pi'_A C\pi'_B Q$ $(a = 1)$ can be computed as:

$$\begin{aligned} p &= \Pr[m = 1-a] = \frac{1}{2}\Pr[m = 1-a|a = 0] + \frac{1}{2}\Pr[m = 1-a|a = 1] \\ &= \frac{1}{2} \mathop{\mathbb{E}}_{([\rho],\tilde{\rho}) \leftarrow \mathcal{AS}_{UBQC1} \vdash^{\sigma} Q} [\mathrm{Tr}((I-\rho)\tilde{\rho})] + \frac{1}{2} \mathop{\mathbb{E}}_{([\rho],\tilde{\rho}) \leftarrow \mathcal{A}\pi'_A C\pi'_B Q} [\mathrm{Tr}(\rho\tilde{\rho})] \\ &\geq \frac{1}{2} - \frac{1}{2}(1-\varepsilon) + \frac{1}{2}(1 - \mathsf{negl}(n)) = \frac{1}{2} + \frac{1}{2}(\varepsilon - \mathsf{negl}(n)) \end{aligned} \tag{6.42}$$

But from Eq.(6.39) we know that the probability to distinguish between $\mathcal{AS}_{UBQC1} \vdash^{\sigma} Q$ and $\mathcal{A}\pi'_A C\pi'_B Q$ is at most $\frac{1}{2}(1 + \mathsf{negl}(n))$.
Therefore, we must have $p \leq \frac{1}{2}(1 + \mathsf{negl}(n))$, which implies that:

$$\begin{aligned} &\frac{1}{2} + \frac{1}{2}(\varepsilon - \mathsf{negl}(n)) \leq \frac{1}{2}(1 + \mathsf{negl}(n)) \\ &\varepsilon \leq 2\mathsf{negl}(n) \end{aligned} \tag{6.43}$$

Consequently, we obtain:

$$\mathop{\mathbb{E}}_{([\rho],\tilde{\rho}) \leftarrow \mathcal{AS}_{UBQC1} \vdash^{\sigma} Q} [\mathrm{Tr}(\rho\tilde{\rho})] \geq 1 - \mathsf{negl}(n) \tag{6.44}$$

which concludes the proof.

□

Now, using our main Theorem 6.2.6 we obtain directly that if $\mathcal{S}_{UBQC1}$ is classically-realizable and RSP with respect to the filter $\vdash^{\sigma}$, then it is also describable:

Corollary 6.3.6. *If $\mathcal{S}_{UBQC1}$ is $\mathsf{negl}(n)$-classically-realizable with respect to filter $\vdash^{c=0}$ then $\mathcal{S}_{UBQC1}$ is $\mathsf{negl}(n)$-describable with respect to the converter $\mathcal{A}$ described above.*

Lemma 6.3.7. *Let $\Omega = \{[\rho_i]\}$ be a set of classical descriptions of density matrices, such that for any 2 states ρ_i, ρ_j whose descriptions $[\rho_i], [\rho_j] \in \Omega$, we have: $\mathrm{Tr}(\rho_i\rho_j) \leq 1 - \eta$.*

We define the following rounding operation, that for any quantum state $\tilde{\rho}$ *rounds* $\tilde{\rho}$ *to the closest* ρ_r*, with* $[\rho_r] \in \Omega$*:*

$$[\rho_r] := Round_{\Omega}([\tilde{\rho}]) := \underset{[\rho_r]\in\Omega}{argmax} \mathrm{Tr}(\tilde{\rho}\rho_r) \tag{6.45}$$

Then, if we consider the random variables $[\rho]$ *and* $[\tilde{\rho}]$ *satisfying:* $[\rho] \in \Omega$ *and* $\underset{[\rho],[\rho']}{\mathbb{E}}[\mathrm{Tr}(\rho\tilde{\rho})] \geq 1-\varepsilon$*, where* $\eta > 6\sqrt{\varepsilon}$*, the following relation must hold:*

$$\Pr_{([\rho],[\tilde{\rho}])}[Round_{\Omega}(\tilde{\rho}) = [\rho]] \geq 1-\sqrt{\varepsilon} \tag{6.46}$$

In particular, if $\varepsilon = \mathsf{negl}(n)$*, and* $\eta \neq 0$ *is a constant,* $\Pr(Round_{\Omega}([\tilde{\rho}]) = [\rho]) \geq 1 - \mathsf{negl}(n)$.

Proof. Using the fact that $\underset{[\rho],[\rho']}{\mathbb{E}}[\mathrm{Tr}(\rho\tilde{\rho})] \geq 1-\varepsilon$, we can apply Markov inequality for the random variable $1-\mathrm{Tr}(\rho\rho')$ and obtain:

$$\begin{aligned}
&\Pr_{([\rho],[\tilde{\rho}])}[1 - Tr(\rho\tilde{\rho}) \geq \sqrt{\varepsilon}] \leq \frac{\mathbb{E}[1-\mathrm{Tr}(\rho\tilde{\rho})]}{\sqrt{\varepsilon}} \\
&\Pr_{([\rho],[\tilde{\rho}])}[Tr(\rho\tilde{\rho}) \leq 1-\sqrt{\varepsilon}] \leq \frac{1-\mathbb{E}[\mathrm{Tr}(\rho\tilde{\rho})]}{\sqrt{\varepsilon}} \leq \frac{\varepsilon}{\sqrt{\varepsilon}} = \sqrt{\varepsilon} \\
&\Pr_{([\rho],[\tilde{\rho}])}[Tr(\rho\tilde{\rho}) \geq 1-\sqrt{\varepsilon}] \geq 1-\sqrt{\varepsilon}
\end{aligned} \tag{6.47}$$

Next, we will prove that if $Tr(\rho\tilde{\rho}) \geq 1-\sqrt{\varepsilon}$, this implies that: $Round_{\Omega}([\tilde{\rho}]) = \rho$. In other words, we will show that if $Tr(\rho\tilde{\rho}) \geq 1-\sqrt{\varepsilon}$ then for any $[\rho_i] \in \Omega$ we have $\mathrm{Tr}(\rho_i\tilde{\rho}) \leq \mathrm{Tr}(\rho\tilde{\rho})$.

By contradiction, let us assume that there exists $[\rho_i] \in \Omega$, with $\rho_i \neq \rho$ such that:

$$\mathrm{Tr}(\rho_i\tilde{\rho}) > \mathrm{Tr}(\rho\tilde{\rho}) \geq 1-\sqrt{\varepsilon}$$

But using the transitivity bound of trace distance from Lemma C.1.4, we also have:

$$\mathrm{Tr}(\rho_i\rho) \geq 1-3(\sqrt{\varepsilon}+\sqrt{\varepsilon}) = 1-6\sqrt{\varepsilon} \tag{6.48}$$

On the other hand, both $[\rho]$ and $[\rho_i]$ belong to Ω, so we also have:

$$Tr(\rho_i\rho) \leq 1-\eta < 1-6\sqrt{\varepsilon}$$

which is a contradiction.

Therefore, using Eq.(6.47), we obtain:

$$\Pr_{([\rho],[\tilde{\rho}])}[Round_{\Omega}([\tilde{\rho}]) = [\rho]] \geq 1-\sqrt{\varepsilon} \tag{6.49}$$

which concludes the proof. □

Lemma 6.3.8. *$\mathcal{S}_{UBQC1}$ cannot be $\mathsf{negl}(n)$-describable with respect to converter $\mathcal{A}$.*

Proof. Assume by contradiction that $\mathcal{S}_{UBQC1}$ is $\mathsf{negl}(n)$-describable. Then there must exist a converter $\mathcal{P}$, whose output is a classical description $[\tilde{\rho}]$ such that:

$$\mathop{\mathbb{E}}_{([\rho],[\tilde{\rho}])\leftarrow\mathcal{A}\mathcal{S}_{UBQC1}\mathcal{P}}[\mathrm{Tr}(\rho\tilde{\rho})] \geq 1-\mathsf{negl}(n) \tag{6.50}$$

In the remaining we are going to use the converters $\mathcal{A}$ and $\mathcal{P}$, together with the ideal resource $\mathcal{S}_{UBQC1}$ to construct a 2-party setting that would achieve signalling, which would complete our contradiction proof. More specifically, we will define a converter $\mathcal{D}$ running on the right interface of the resource $\mathcal{S}_{UBQC1}$, which will succeed in recovering the input ϕ_0 chosen at random by $\mathcal{A}$.

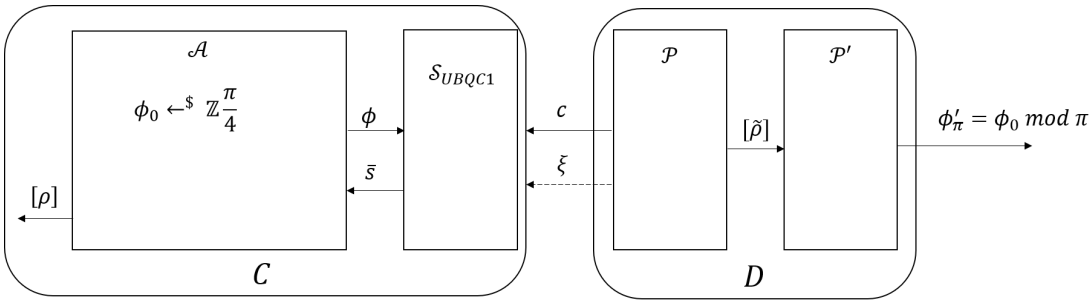


Figure 6.11: Construction of the signalling game between C and D, where the input of C is ϕ_0 and the output of D is $\phi'_\pi = \phi_0 \bmod \pi$.

As shown in Figure 6.11, by defining C as $C := \mathcal{A}\mathcal{S}_{UBQC1}$ and D the converter $D := \mathcal{P}\mathcal{P}'$, where $\mathcal{P}'$ will be formally defined later, then the game between the 2 players can be described as follows: C chooses a random $\phi_0 \in \mathbb{Z}_{\frac{\pi}{4}}$ and D needs to output $\phi_0 \bmod \pi$. This is however impossible, as no message is sent from $\mathcal{S}_{UBQC1}$ to its right interface (as can be observed in Figure 6.11), and thus no message is sent from C to D. Consequently, guessing $\phi_0 \bmod \pi$ is forbidden by the no-signalling principle [GRW80].

Next we need to construct the converter $\mathcal{P}'$ allowing D to give the desired output described above. First we define the set of classical descriptions $\Omega := \{[|+_\theta\rangle\langle+_\theta|] \,|\, \theta \in \{0, \frac{\pi}{4}, \cdots, \frac{7\pi}{4}\}\}$. For simplicity we will denote $[|+_\theta\rangle\langle+_\theta|]$ by $[\theta]$.

Then, we can define $\mathcal{P}'$ as:

- Given $[\tilde{\rho}]$ received from the $\mathcal{P}$, compute $[\tilde{\phi}] := Round_\Omega([\tilde{\rho}])$;
- Output $\phi'_\pi := \tilde{\phi} \bmod \pi$.

To show that D can win the above game, we now need to prove that the output of $\mathcal{P}'$, ϕ'_π, is equal to $\phi_0 \bmod \pi$ with overwhelming probability.

Let us first examine the first operation of $\mathcal{P}'$. Consider 2 different state descriptions from Ω: $[\rho_i] = [\theta_i]$ and $[\rho_j] = [\theta_j]$. Then we have:

$$\begin{aligned} \mathrm{Tr}\big(\rho_i\rho_j\big) &= \mathrm{Tr}\big(|+_{\theta_i}\rangle\langle+_{\theta_i}|\,|+_{\theta_j}\rangle\langle+_{\theta_j}|\big) \\ &= \mathrm{Tr}\big(|\langle+_{\theta_i}|+_{\theta_j}\rangle|^2\big) = |\langle+_{\theta_i}|+_{\theta_j}\rangle|^2 \\ &= \frac{1}{4}|1+e^{i(\theta_j-\theta_i)}|^2 = \frac{1}{2}(1+cos(\theta_i-\theta_j)) \end{aligned} \tag{6.51}$$

Therefore, there exists a constant $\eta > 0$ such that: $\mathrm{Tr}\big(\rho_i\rho_j\big) \leq 1-\eta$. Additionally, from Eq.(6.50), we have that $\mathcal{S}_{UBQC1}$ is ε-describable with $\varepsilon = \mathsf{negl}(n)$, hence we also have (for sufficiently large n) that $\eta > 6\sqrt{\varepsilon}$. Consequently, by applying Lemma 6.3.7 we have that:

$$\Pr_{([\rho],[\tilde{\rho}])\leftarrow\mathcal{A}\mathcal{S}_{UBQC1}\mathcal{P}}\left[Round_\Omega([\tilde{\rho}]) = [\rho]\right] \geq 1-\sqrt{\varepsilon} = 1-\mathsf{negl}(n) \tag{6.52}$$

But using the definition of converter $\mathcal{A}$ we have that $[\rho] = [\phi']$ and $\phi' = \phi_0 + \bar{s}\pi$ and hence $\phi' \bmod \pi = \phi_0 \bmod \pi$. Finally, by using the definition of $\mathcal{P}'$, the Eq.(6.52) becomes equivalent to:

$$\Pr_{([\rho],\phi'_\pi)\leftarrow\mathcal{A}\mathcal{S}_{UBQC1}\mathcal{P}\mathcal{P}'}\left[\phi'_\pi = \phi_0 \bmod \pi\right] \geq 1-\mathsf{negl}(n) \tag{6.53}$$

Putting things together, we have obtained a game between $C = \mathcal{A}\mathcal{S}_{UBQC1}$ and $D = \mathcal{P}\mathcal{P}'$, where as explained above, C picks a random $\phi_0 \in \mathbb{Z}_{\frac{\pi}{4}}$ and D needs to output $\phi_0 \bmod \pi$. From Eq.(6.53), we have obtained that D can win this game with overwhelming probability, however, since there is no information transfer from C to D, winning this game with probability better than $1/4$ (guessing the 2 bits of $\phi_0 \bmod \pi$ uniformly at random) would imply signalling.

□

6.3.2 Impossibility of Composable General $\mathsf{UBQC}_{\mathsf{CC}}$

From Theorem 6.3.2, we know that it is impossible to implement a *composable classical client* UBQC protocol performing a computation consisting of a single qubit. In this last section, we prove that this no-go result generalizes to the impossibility of $\mathsf{UBQC}_{\mathsf{CC}}$ on computations using an arbitrary number of qubits. To show this we will reduce the general setting to the single-qubit case.

Definition 6.3.9 (Ideal Resource general UBQC)**.** *The ideal resource $\mathcal{S}_{UBQC}$ achieves blind quantum computation, where the computation is specified by a set of angles $\{\phi_{i,j}\}_{i\in\{1,\cdots,n\},j\in\{1,\cdots,m\}}$. The ideal resource is defined similar to $\mathcal{S}_{UBQC1}$ with the difference that in the honest run, the resource performs an MBQC computation described by angles $\{\phi_{i,j}\}_{i,j}$ and the output on the left interface represents the classical outcome of the computation.*

Theorem 6.3.10 (No-go Composable Classical-Client UBQC)**.** *Let (P_A,P_B) be a protocol interacting only through a classical channel C such that $(\theta,\rho_B)\leftarrow(P_ACP_B)$ where $\theta\in\mathbb{Z}_{\frac{\pi}{4}}$ and such that the trace distance between ρ_B and $|+_\theta\rangle\langle+_\theta|$ is negligible with overwhelming probability.*

If we define (π_A^G,π_B^G) as the UBQC protocol on any quantum computation (described by a graph G), that uses (P_A,P_B) as a sub-protocol to replace the quantum channel, then (π_A^G,π_B^G) is not composable secure, i.e. there exists no simulator such that:

$$\begin{aligned}&\pi_A^G C\pi_B^G \approx_\varepsilon \mathcal{S}_{UBQC}\vdash^{c=0}\\&\pi_A^G C\approx_\varepsilon \mathcal{S}_{UBQC}\sigma\end{aligned}\tag{6.54}$$

for $\varepsilon=\mathsf{negl}(n)$.

Proof. We will show that we can reduce this to the setting of a single qubit, specifically where the computation is described by a single angle ϕ (Theorem 6.3.2).

As the graph G describing the target computation consists of at least one output qubit, we will denote by w the index of the last output qubit. The main idea is to consider a distinguisher between the real and ideal world that would conveniently choose the client's input in the following way: for every node $i\in G$, we choose $\phi_i:=0$ if $i\neq w$ and $\phi_i:=\phi$ if $i=w$.

Then on the right interface (server's side), the distinguisher will behave like the honest π_B^G, except that he will not entangle the qubits produced by the sub-protocol (P_A,P_B). Finally, for the output qubit ρ_w, instead of measuring it, he will send $s:=0$ on the left interface, and he will rotate the qubit with angle $-\delta_w$, $\tilde{\rho_w}=R_Z(-\delta_w)\rho_w$ and will output $\tilde{\rho_w}$, exactly as π_B' in the single-qubit case.

Then it is easy to observe that for every qubit $i\neq w$, we have that the angle δ_i (received by distinguisher from the right interface corresponding to the client) is: $\delta_i:=\theta_i+r_i\pi$ (as the qubits are not entangled and $\phi_i=0$). As a result, the distinguisher by measuring $|+_{\theta_i}\rangle$ in the basis $\{|+_\delta\rangle,|+_\delta\rangle\}$, obtains: $M_ZHR_Z(-\delta_i)|+_{\theta_i}\rangle=M_ZH|_{r_i\pi}\rangle=|r_i\rangle$.

Therefore, we also have that $\bar{s}_i = r_i \oplus r_i = 0$. Additionally, for all nodes i, including w, we have that the dependency from previous measurements s_i^X and s_i^Z will be: $s_i^X = \oplus_{j \in D_i^X} \bar{s}_j = 0$ and $s_i^Z = \oplus_{j \in D_i^Z} \bar{s}_j = 0$ (where D_i^X and D_i^Z are the X and Z dependency sets, as described in Protocol 7). Therefore, for the output node w we have that the angle received by distinguisher is:

$$\delta_w = \theta_w + (-1)^{s_w^X} \phi_w + s_w^Z \pi + r_w \pi = \theta_w + \phi + r_w \pi \tag{6.55}$$

Consequently, we have reached exactly the single-qubit setting (as shown in Figure 6.9, where the computation is described by a single angle ϕ, which we have proved is impossible. □

Chapter 7

Conclusions

The current dissertation studies the problem of secure delegation of quantum computations between a fully classical honest client and a quantum untrusted server (CSDQC). Our results can be summarizes as follows:

- Chapter 3 ([ACGK19]): We show that achieving information-theoretic secure delegation of quantum computations between a fully classical client and a quantum server is implausible, by showing certain complexity theoretic implications.
- Chapter 4 ([CCKW18]): We provide a solution for the CSDQC problem under post-quantum computational security. Our solution is based on constructing a remote state preparation primitive. However, this first candidate is only secure in the honest-but-curious framework.
- Chapter 5 ([CCKW19]): We present a second construction for a remote state preparation with improved security, namely achieving security against malicious adversary. Both constructions rely on the same cryptographic primitive of trapdoor functions, which in turn rely on the learning-with-errors problem.
- Chapter 6 ([BCC$^+$20]): We examine the composability property of the remote state preparation primitive in a general context, but also when used as a submodule for the task of secure delegation of quantum computations.

Given the need for a practical solution for the CSDQC problem in order to exploit the full potential of quantum computers, a key next step is represented by the optimisation of the proposed schemes and bringing our theoretical results close to practice. Secondly, another future work would be investigating the use of our proposed primitive - the classical remote state preparation - as a sub-protocol in different communication and computation protocols, such as in the quantum multiparty computation setting [CCKM20].

Appendix A

App: HBC – QFactory

A.1 Full proof of Theorem 4.4.4

Proof. From Eq. (4.12) we have the definition of $\widetilde{B}$ in terms of the three corresponding bits and we aim to prove that it is hard-core, i.e. that Eq. (4.14) is satisfied. We will follow the five steps outlined in the main text. Before that let us define some simple identities that will be used. For any $a,b,d,e \in \mathbb{N}$, we have:

$$(a+b) \bmod 8 = (a \bmod 8 + b \bmod 8) \bmod 8 \tag{I1}$$

$$[(a+b) \bmod 8] \bmod 4 = (a \bmod 4 + b \bmod 4) \bmod 4 \tag{I2}$$

$$[(a+b) \bmod 4] \bmod 2 = (a \bmod 2 + b \bmod 2) \bmod 2 \tag{I3}$$

$$(2a) \bmod 4 = 2 \cdot (a \bmod 2) \tag{I4}$$

$$(2a) \bmod 8 = 2 \cdot (a \bmod 4) \tag{I5}$$

$$(2d+e) \bmod 4 - e \bmod 2 = [2d+e-(e \bmod 2)] \bmod 4 \tag{I6}$$

$$(2d+e) \bmod 8 - e \bmod 2 = [2d+e-(e \bmod 2)] \bmod 8 \tag{I7}$$

We now return to Eq. (4.12):

$$\widetilde{B} = g(x-x') = \sum_{i=1}^{n} (x_i - x'_i)(4b_i + \alpha_i) \bmod 8$$

where $\widetilde{B} = \widetilde{B_1}\widetilde{B_2}\widetilde{B_3}$, with $\widetilde{B_j} \in \{0,1\}$. We also define $\widetilde{x} = x \oplus x' \in \{0,1\}^n$ and $z \in \{-1,0,1\}^n$ be the vector defined as: $z_i = x_i - x'_i = (-1)^{x'_i}\widetilde{x_i}, \forall i \in \{1,2,...,n\}$.

Step 1: We will rewrite this expression in terms of single bits and obtain the expression of Eq. (4.14). We have $g(z) = \sum_{i=1}^{n} z_i(4b_i + \alpha_i) \bmod 8$, or equivalently:

$$4\widetilde{B_1} + 2\widetilde{B_2} + \widetilde{B_3} = \left[\left(4\sum_{i=1}^{n} z_i b_i \right) \bmod 8 + \left(\sum_{i=1}^{n} z_i \alpha_i \right) \bmod 8 \right] \bmod 8$$

We define the following terms: $\alpha_i = 4\alpha_i^{(1)} + 2\alpha_i^{(2)} + \alpha_i^{(3)}$, where $\alpha_i^{(1)}, \alpha_i^{(2)}, \alpha_i^{(3)}$ are the 3 bits of α_i and $\alpha^{(j)} \in \{0,1\}^n$ are the vectors consisting of the j-th bit of all values α_i, $\forall i \in \{1,2,...,n\}, j \in \{1,2,3\}$;

$$S_0 = \sum_{i=1}^{n} z_i b_i \quad ; \quad S_1 = \sum_{i=1}^{n} z_i \alpha_i^{(1)}$$
$$S_2 = \sum_{i=1}^{n} z_i \alpha_i^{(2)} \quad ; \quad S_3 = \sum_{i=1}^{n} z_i \alpha_i^{(3)}$$

We also notice that under $\bmod 2$, we have that:

$$S_j \bmod 2 = \sum_{i=1}^{n} \widetilde{x_i} \alpha_i^{(j)} \bmod 2 = \langle \widetilde{x}, \alpha^{(j)} \rangle \bmod 2 \text{ , for } j \in \{1,2,3\}.$$

Then, we have:

$$4\widetilde{B_1} + 2\widetilde{B_2} + \widetilde{B_3} = \left(\sum_{i=1}^{n} (x_i - x_i')(4b_i + \alpha_i) \right) \bmod 8 =$$
$$\left[4S_0 + \sum_{i=1}^{n} (x_i - x_i')(4\alpha_1^{(i)} + 2\alpha_2^{(i)} + \alpha_3^{(i)}) \right] \bmod 8$$
$$4\widetilde{B_1} + 2\widetilde{B_2} + \widetilde{B_3} = (4S_0 + 4S_1 + 2S_2 + S_3) \bmod 8 \tag{A.1}$$

Applying $\bmod 2$ to Eq. (A.1), we get:

$$\widetilde{B_3} = (4S_0 \bmod 2 + 4S_1 \bmod 2 + 2S_2 \bmod 2 + S_3 \bmod 2) \bmod 2$$
$$\boxed{\widetilde{B_3} = S_3 \bmod 2 = \langle \widetilde{x}, \alpha^{(3)} \rangle \bmod 2} \tag{A.2}$$

If, instead we apply $\bmod 4$ to Eq. (A.1), we get:

$$\begin{aligned} 2\widetilde{B_2} + \widetilde{B_3} &= [4S_0 \bmod 4 + 4S_1 \bmod 4 + 2S_2 \bmod 4 + S_3 \bmod 4] \bmod 4 \\ 2\widetilde{B_2} + \widetilde{B_3} &= [(2S_2) \bmod 4 + S_3 \bmod 4] \bmod 4. \text{ Using I4, we have:} \\ 2\widetilde{B_2} + \widetilde{B_3} &= [2(S_2 \bmod 2) + S_3 \bmod 4] \bmod 4 \end{aligned}$$

$$\widetilde{B_2} = \frac{1}{2}\{[2(S_2 \bmod 2) + S_3 \bmod 4] \bmod 4 - S_3 \bmod 2\}. \text{ Using I6:}$$

$$\widetilde{B_2} = \frac{1}{2}[2(S_2 \bmod 2) + S_3 \bmod 4 - S_3 \bmod 2] \bmod 4$$

$$\widetilde{B_2} = \frac{1}{2}\left\{2 \cdot \left[(S_2 \bmod 2) + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right]\right\} \bmod 4. \text{ Using I4, we obtain:}$$

$$\widetilde{B_2} = \left[S_2 \bmod 2 + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right] \bmod 2.$$

$$\widetilde{B_2} = S_2 \bmod 2 \oplus \left(\frac{S_3 \bmod 4 - S_3 \bmod 2}{2}\right)$$

$$\boxed{\widetilde{B_2} = \langle \widetilde{x}, \alpha^{(2)} \rangle \bmod 2 \oplus \left(\frac{S_3 \bmod 4 - S_3 \bmod 2}{2}\right)} \tag{A.3}$$

Finally, we can derive $\widetilde{B_1}$:

$$\widetilde{B_1} = \tfrac{1}{4}\{(4S_0 + 4S_1 + 2S_2 + S_3) \bmod 8 - (S_3 \bmod 2) - 2\left[\left(S_2 \bmod 2 + \tfrac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\}$$

Using I7:

$$\widetilde{B_1} = \tfrac{1}{4}\{(4S_0 + 4S_1 + 2S_2 + S_3 - (S_3 \bmod 2)) \bmod 8 - 2\left[\left(S_2 \bmod 2 + \tfrac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\}$$

Using I5:

$$\begin{aligned}\widetilde{B_1} &= \frac{1}{4}\left\{2\left[\left(2S_0 + 2S_1 + S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 4\right] - \right. \\ &\left. -2\left[\left(S_2 \bmod 2 + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\right\} \\ &= \frac{1}{2}\left[\left(2S_0 + 2S_1 + S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 4 - \right. \\ &\left. - \left(S_2 \bmod 2 + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\end{aligned}$$

Using I6 we can rewrite the first term, and we get:

$$\widetilde{B_1} = \frac{1}{2}\left\{\left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2 + \left[2(S_0 + S_1) + S_2 + \frac{S_3 - S_3 \bmod 2}{2} - \right.\right.$$
$$\left.\left. - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2\right] \bmod 4 - \left[S_2 \bmod 2 + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right] \bmod 2\right\}$$

Combining the first and third term:

$$\begin{aligned}\widetilde{B_1} &= \frac{1}{2}\left\{\left[(S_2 - S_2 \bmod 2) + \frac{S_3 - S_3 \bmod 4}{2}\right] \bmod 2 + \right. \\ &\left. + \left[2(S_0 + S_1) + S_2 + \frac{S_3 - S_3 \bmod 2}{2} - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2\right] \bmod 4\right\}\end{aligned}$$

We notice that both $S_2 - S_2 \bmod 2$ and $\frac{S_3 - S_3 \bmod 4}{2}$ are even, so the first big term is 0:

$$\widetilde{B_1} = \frac{1}{2}\left\{\left[2(S_0+S_1)+S_2+\frac{S_3-S_3 \bmod 2}{2}-\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right) \bmod 2\right] \bmod 4\right\}$$

which can be rewritten as:

$$\widetilde{B_1} = \frac{1}{2}\left\{\left\{2\cdot\left[S_0+S_1+\frac{\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right)-\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right) \bmod 2}{2}\right]\right\} \bmod 4\right\}$$

Finally using I4, we get:

$$\widetilde{B_1} = \left[S_0+S_1+\frac{\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right)-\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2$$

$$\widetilde{B_1} = S_1 \bmod 2 \oplus S_0 \bmod 2 \oplus \left[\frac{\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right)-\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2$$

$$\begin{aligned}\widetilde{B_1} = {} & \langle \widetilde{x}, \alpha^{(1)} \rangle \bmod 2 \oplus \langle \widetilde{x}, b \rangle \bmod 2 \oplus \\ & \oplus \left[\frac{\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right)-\left(S_2+\frac{S_3-S_3 \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2\end{aligned} \tag{A.4}$$

Important observation: $\widetilde{B_1}$, $\widetilde{B_2}$, $\widetilde{B_3}$ all depend on the same value of x and x' (or $\widetilde{x}$, or z). Therefore, to make our analysis easier, we can consider that z and $\widetilde{x}$ are fixed. Then, if we define the function:

$$B(r) = \langle \widetilde{x}, r \rangle \bmod 2 = \left(\sum_{i=1}^{n} \widetilde{x}_i r_i\right) \bmod 2 \tag{A.5}$$

we can rewrite $\widetilde{B_3}, \widetilde{B_2}, \widetilde{B_1}$ as in Eq. (4.14) completing Step 1:

$$\begin{aligned}\widetilde{B_3} &= B\left(\alpha^{(3)}\right) \\ \widetilde{B_2} &= B\left(\alpha^{(2)}\right) \oplus h_2(z, \alpha^{(3)}) \\ \widetilde{B_1} &= B\left(\alpha^{(1)}\right) \oplus h_1(z, \alpha^{(3)}, \alpha^{(2)}, b)\end{aligned} \tag{A.6}$$

Where:

$$h_2(z, \alpha^{(3)}) := \frac{\langle z, \alpha^{(3)} \rangle \bmod 4 - \langle z, \alpha^{(3)} \rangle \bmod 2}{2} \tag{A.7}$$

$$\begin{aligned}h_1(z, \alpha^{(3)}, \alpha^{(2)}, b) := {} & \langle z, b \rangle \bmod 2 \oplus \\ & \oplus \left[\frac{\left(\langle z, \alpha^{(2)} \rangle + \frac{\langle z, \alpha^{(3)} \rangle - \langle z, \alpha^{(3)} \rangle \bmod 2}{2}\right) - \left(\langle z, \alpha^{(2)} \rangle + \frac{\langle z, \alpha^{(3)} \rangle - \langle z, \alpha^{(3)} \rangle \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2\end{aligned} \tag{A.8}$$

Step 2: We see from Eq. (A.4) that each of the three bits involve a term similar to that of the GL theorem 2.3.10 (the $B\left(\alpha^{(i)}\right)$ term), but with two the important differences. First, there is another term, and the bits of $\widetilde{B}$ are XORs of the GL-looking term and that other one. The second type of terms (that involve h_1, h_2) depend on variables that appear in the expressions of other bits, potentially introducing correlations among the different bits. We will deal with the issue of correlations in Step 3, while with the effects of having extra terms in Steps 4 and 5. Here we deal with the second important difference, namely that the GL-looking terms (those of the form $\langle \tilde{x}, r \rangle$ mod 2) depend on $\tilde{x}$ rather than x in the inner product. For the remaining Step 2, we assume that the first issue is resolved and it all reduces to GL theorem subject to having $\tilde{x}$ rather than x.

Since we have $\tilde{x}$ in our expression if we follow the same proof with that of the GL theorem we can follow the proof until the point that we end up with obtaining a polynomial number of guesses for $\tilde{x}$ of which one is the correct value with probability negligibly close to unity. Now to continue with the proof we are lacking two elements. First, in GL theorem they use the fact that computing $f(x)$ given x is easy, and check one-by-one the polynomial guesses to see which one (if any) is correct. We cannot do this since we only obtain $\tilde{x}$ and there is no way with no extra information to check if $\tilde{x}$ actually corresponds to a given image $y = f(x) = f(x')$. The second issue, is that even if we could check this, having obtained $\tilde{x}$ does not contradict the definition of one-way function (definition 2.3.2).

We resolve both these issues with two observations.

Observation 1: We notice that because of the 2-regularity property of f, $\widetilde{x}$ is uniquely determined by x ($f(x) = f(x')$, $\widetilde{x} = x \oplus x'$).

Observation 2: The assumption that our 2-regular trapdoor function f is second preimage resistant (i.e. a QPT adversary given x, cannot find the second preimage x', where $f(x) = f(x')$) means that:

$$\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x' \text{ such that } f(x) = f(x')] \leq \mathsf{negl}(n)$$

As

$$\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x'] = \Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x' \oplus x]$$

we have that:

$$\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = \widetilde{x}] \leq \mathsf{negl}(n) \tag{A.9}$$

As we have mentioned, following the GL theorem proof we would obtain polynomially many guesses $\tilde{x}_g$ for $\tilde{x}$ (where subscript g stands for guess). Now by the second preimage resistance, if we are given x we should be unable to obtain x' in polynomial time. However, using our polynomially many guesses for $\tilde{x}$ and checking for each guess if $f(x \oplus \tilde{x}_g) = f(x)$ we can obtain with probability negligible close to unity the correct $\tilde{x}$ and therefore come to contradiction with Eq. (A.9).

Step 3: Since the different bits involve common variables, to prove that our function is hard-core we need to consider the issue of correlations. One way to deal with this would be to prove the independence of both the bits and of the optimal guessing algorithms. We, instead, use the *Vazirani-Vazirani Theorem* 2.3.11, which for our case it means that it suffices to show that: $\widetilde{B_1}$, $\widetilde{B_2}$, $\widetilde{B_3}$, $\widetilde{B_1} \oplus \widetilde{B_2}$, $\widetilde{B_1} \oplus \widetilde{B_3}$, $\widetilde{B_2} \oplus \widetilde{B_3}$, $\widetilde{B_1} \oplus \widetilde{B_2} \oplus \widetilde{B_3}$ are all hard-core predicates for f.

The most general expression that captures all these hard-core predicates (formed from the subsets of $\{\widetilde{B_1}, \widetilde{B_2}, \widetilde{B_1}\}$) is:

$$E(x, r_1, r_2, r_3) = \langle \widetilde{x}, r_1 \oplus r_2 \rangle \bmod 2 \oplus g(z, r_2, r_3) \tag{A.10}$$

where g can be any binary function. Using $\langle \widetilde{x}, r_1 \oplus r_2 \rangle \bmod 2 = \langle \widetilde{x}, r_1 \rangle \bmod 2 \oplus \langle \widetilde{x}, r_2 \rangle \bmod 2$ we can rewrite this as:

$$E(x, r_1, r_2, r_3) = \langle \widetilde{x}, r_1 \rangle \bmod 2 \oplus g'(z, r_2, r_3) \tag{A.11}$$

where $g'(z, r_2, r_3) = \langle \widetilde{x}, r_2 \rangle \bmod 2 \oplus g(z, r_2, r_3)$. In other words, in order to prove that $\widetilde{B_1}\widetilde{B_2}\widetilde{B_3}$ is a hard-core function for f, it suffices to prove that $E(x, r_1, r_2, r_3)$ is a hard-core predicate for f.

Step 4: In this step, we will see how we can effectively fix all but one variables, and turn Eq. (A.11) to depend only on r_1.

We want to prove that if there exists a QPT algorithm $\mathcal{A}$ that can guess the predicate E as given in Eq. (A.11), $\mathcal{A}(f(x), r_1, r_2, r_3) = E(x, r_1, r_2, r_3)$ with probability non-negligible better than $1/2$, then the second preimage resistance assumption is violated by constructing a QPT algorithm $\mathcal{A}'$ that, when given x can obtain $\widetilde{x}$, $\mathcal{A}'(f(x), 1^n) = \widetilde{x}$, with non-negligible probability.

We now assume that the advantage $\mathcal{A}$ has in computing is $\varepsilon(n)$, without restricting $\varepsilon(n)$ to be non-negligible, aiming to reach a contradiction if this $\varepsilon(n)$ is inverse polynomial. We therefore assume:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n\\ r_1\leftarrow\{0,1\}^n\\ r_2\leftarrow\{0,1\}^n\\ r_3\leftarrow\{0,1\}^n}}[\mathcal{A}(f(x),r_1,r_2,r_3)=E(x,r_1,r_2,r_3)]=\frac{1}{2}+\varepsilon(n) \tag{A.12}$$

Since the different variables (x,r_1,r_2,r_3) are chosen randomly and independently we can effectively "fix" one variable. We can consider the set of values of that variable that satisfy some condition that we need and name these values "Good" values (e.g. the guessing algorithm $\mathcal{A}$ to succeed with higher than negligible probability). Then we can work with the assumption that the fixed variable is within the "Good" set, with only caveat that at the end, whatever probability of inversion we obtain, is conditional on the fixed variables being "Good" and thus we need to multiply that probability with the probability that the fixed variable is "Good". For this reason, it is important that the probability of being "Good" (ratio between cardinality of Good values and total number of values) should be at least inverse polynomial.

We will, therefore, be using the following Lemma:

Lemma A.1.1. *Let* $\Pr_{(v_1,\cdots,v_k)\leftarrow\{0,1\}^n\times\cdots\times\{0,1\}^n}[Guessing]\geq p+\varepsilon(n)$*, then for any variable* v_i*, there exists a set* $Good_{v_i}\subseteq\{0,1\}^n$ *of size at least* $\frac{\varepsilon(n)}{2}2^n$*, such that for all* $v_i\in Good_{v_i}$*, we have:*

$$\Pr_{(v_1,\cdots,\not{v_i},\cdots,v_k)\leftarrow\{0,1\}^n\times\cdots\times\{0,1\}^n}[Guessing]\geq p+\frac{\varepsilon(n)}{2}$$

where the latter probability is taken over all variables except v_i*.*

Proof.

$$\begin{aligned}p+\varepsilon(n)&\leq \tfrac{1}{2^n}\textstyle\sum_{v_i\in \mathrm{Good}_{v_i}}\Pr_{(v_1,\cdots,\not{v_i},\cdots,v_k)|v_j\in\{0,1\}^n}[\mathrm{Guessing}]+\\ &\quad \tfrac{1}{2^n}\textstyle\sum_{v_i\notin \mathrm{Good}_{v_i}}\Pr_{(v_1,\cdots,\not{v_i},\cdots,v_k)|v_j\in\{0,1\}^n}[\mathrm{Guessing}]\\ &\leq \tfrac{1}{2^n}|\mathrm{Good}_{v_i}|+\tfrac{1}{2^n}\textstyle\sum_{v_i\notin \mathrm{Good}_{v_i}}(p+\frac{\varepsilon(n)}{2})\end{aligned} \tag{A.13}$$

$$\begin{aligned}p+\varepsilon(n)&\leq \frac{1}{2^n}|\mathrm{Good}_{v_i}|+(p+\frac{\varepsilon(n)}{2})\\ \frac{\varepsilon(n)}{2}2^n&\leq |\mathrm{Good}_{v_i}|\end{aligned} \tag{A.14}$$

□

Now we return to Eq. (A.12), we fix the set of Good_x, the set of inputs x such that:

$$\Pr_{\substack{r_1\leftarrow\{0,1\}^n\\ r_2\leftarrow\{0,1\}^n\\ r_3\leftarrow\{0,1\}^n}}[\mathcal{A}(f(x),r_1,r_2,r_3)=E(x,r_1,r_2,r_3)]\geq\frac{1}{2}+\frac{\varepsilon(n)}{2}\quad\forall x\in\text{Good}_x \tag{A.15}$$

and using Lemma A.1.1 we have $|\text{Good}_x|\geq\frac{\varepsilon(n)}{2}2^n$. Note that fixing x is equivalent with fixing $\tilde{x}$ or z, given the definition of the 2-regular function f. Starting with Eq. (A.15) we can now fix r_3 (conditional on $x\in\text{Good}_x$):

$$\Pr_{\substack{r_1\leftarrow\{0,1\}^n\\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}(f(x),r_1,r_2,r_3)=E(x,r_1,r_2,r_3)]\geq\frac{1}{2}+\frac{\varepsilon(n)}{4}$$
$$\forall x\in\text{Good}_x\wedge\forall r_3\in\text{Good}_{r_3} \tag{A.16}$$

where using again Lemma A.1.1 we have $|\text{Good}_{r_3}|\geq\frac{\varepsilon(n)}{4}2^n$. Finally, we can fix r_2 (conditional on $x\in\text{Good}_x$ and $r_3\in\text{Good}_{r_3}$):

$$\Pr_{r_1\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r_1,r_2,r_3)=E(x,r_1,r_2,r_3)]\geq\frac{1}{2}+\frac{\varepsilon(n)}{8}$$
$$\forall x\in\text{Good}_x\wedge\forall r_3\in\text{Good}_{r_3}\wedge\forall r_2\in\text{Good}_{r_2} \tag{A.17}$$

and again by Lemma A.1.1 we have $|\text{Good}_{r_2}|\geq\frac{\varepsilon(n)}{8}2^n$.

Step 5: In Eq. (A.17) the only variable is r_1. Using Eq. (A.11) we can see that given that x,r_2,r_3 are all fixed, $E(x,r_1,r_2,r_3)=\langle\tilde{x},r_1\rangle\oplus g'(z,r_2,r_3)$ where $g'(z,r_2,r_3)=c$ is constant. Because c is a constant, we can define $\tilde{\mathcal{A}}=\mathcal{A}\oplus c$. Now, we can easily see that:

$$\Pr_{r_1\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r_1,r_2,r_3)=\langle\widetilde{x},r_1\rangle \bmod 2\oplus g'(z,r_2,r_3)]$$
$$=\Pr_{r_1\leftarrow\{0,1\}^n}[\tilde{\mathcal{A}}(f(x),r_1,r_2,r_3)=\langle\widetilde{x},r_1\rangle \bmod 2]$$

So, using Eq. (A.17), we obtain

$$\Pr_{r_1\leftarrow\{0,1\}^n}[\tilde{\mathcal{A}}(f(x),r_1,r_2,r_3)=\langle\widetilde{x},r_1\rangle \bmod 2]\geq\frac{1}{2}+\frac{\varepsilon(n)}{8}$$
$$\forall x\in\text{Good}_x\wedge\forall r_3\in\text{Good}_{r_3}\wedge\forall r_2\in\text{Good}_{r_2} \tag{A.18}$$

which is exactly the expression in GL theorem. There, one obtains guesses for inversion, i.e. to obtain $\tilde{x}$ with a polynomial in $\varepsilon(n)$ probability of success, given the fixed x, r_2, r_3's. Multiplying this with the probability of actually being in Good_x and Good_{r_3} and Good_{r_2} we obtain another polynomial in $\varepsilon(n)$. This rules out the possibility of $\varepsilon(n)$ being inverse polynomial, since that would break the second preimage resistance. As we have already stated, guessing $\tilde{x}$ with inverse polynomial success probability does not contradict the one-way property of the trapdoor function, but it does contradict the second preimage resistance, since given x and $\tilde{x}$ one can obtain deterministically x'.

Concretely, using GL proof to construct from $\tilde{\mathcal{A}}$, a QPT algorithm $\mathcal{A}'$ that obtains $\widetilde{x}$, $\mathcal{A}'(f(x), 1^n) = \widetilde{x}$ for all inputs $x \in \{0,1\}^n$, when, $x \in \text{Good}_x$, $r_3 \in \text{Good}_{r_3}$ and $r_2 \in \text{Good}_{r_2}$, this algorithm $\mathcal{A}'$ succeeds with probability:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}} [\mathcal{A}'(f(x), 1^n) = \widetilde{x} | x \in \text{Good}_x \wedge r_3 \in \text{Good}_{r_3} \wedge r_2 \in \text{Good}_{r_2}]$$

$$\geq \frac{\varepsilon^2(n)}{2(32n + \varepsilon^2(n))} \geq \frac{\varepsilon^2(n)}{2(32n+1)}$$

Then, we have:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}} [\mathcal{A}'(f(x), 1^n) = \widetilde{x}] \geq$$

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}} [\mathcal{A}'(f(x), 1^n) = \widetilde{x} \wedge x \in \text{Good}_x \wedge r_3 \in \text{Good}_{r_3} \wedge r_2 \in \text{Good}_{r_2}] \geq$$

$$\geq \Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}} [\mathcal{A}'(f(x), 1^n) = \widetilde{x} | x \in \text{Good}_x \wedge r_3 \in \text{Good}_{r_3} \wedge r_2 \in \text{Good}_{r_2}] \cdot$$

$$\cdot \Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}} [x \in \text{Good}_x \wedge r_3 \in \text{Good}_{r_3} \wedge r_2 \in \text{Good}_{r_2}]$$

We now see that:

$$\Pr[x \in \text{Good}_x \wedge r_3 \in \text{Good}_{r_3} \wedge r_2 \in \text{Good}_{r_2}] =$$

$$\Pr[r_2 \in \text{Good}_{r_2} | r_3 \in \text{Good}_{r_3} \wedge x \in \text{Good}_x] \cdot \Pr[r_3 \in \text{Good}_{r_3} | x \in \text{Good}_x] \times \Pr[x \in \text{Good}_x]$$

By construction we have $\Pr[x \in \text{Good}_x] = \frac{|\text{Good}_x|}{2^n}$ and $\Pr[r_3 \in \text{Good}_{r_3} | x \in \text{Good}_x] = \frac{|\text{Good}_{r_3}|}{2^n}$ and $\Pr[r_2 \in \text{Good}_{r_2} | r_3 \in \text{Good}_{r_3} \wedge x \in \text{Good}_x] = \frac{|\text{Good}_{r_2}|}{2^n}$, which leads to

$$\begin{aligned}\Pr_{\substack{x\leftarrow\{0,1\}^n\\ r_3\leftarrow\{0,1\}^n\\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n)=\widetilde{x}] &\geq \frac{\varepsilon^2(n)}{2(32n+1)}\cdot\frac{|\mathrm{Good}_x|}{2^n}\cdot\frac{|\mathrm{Good}_{r_3}|}{2^n}\cdot\frac{|\mathrm{Good}_{r_2}|}{2^n}\\ &\geq \frac{\varepsilon^5(n)}{128(32n+1)} \qquad (A.19)\end{aligned}$$

where we can view r_3 and r_2 as the internal randomness of the inversion algorithm $\mathcal{A}'$. It is clear that if $\varepsilon(n)$ is non-negligible, it means that there exists polynomial $p(n)$ such that $\varepsilon(n)=1/p(n)$, and then

$$\Pr_{\substack{x\leftarrow\{0,1\}^n\\ r_3\leftarrow\{0,1\}^n\\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n)=\widetilde{x}] \geq \frac{1}{128(32n+1)p(n)^5} \qquad (A.20)$$

which as explained in Step 2 breaks second preimage resistance, Eq. (A.9). Since all the terms given in Step 3 ($\widetilde{B}_i, \widetilde{B}_i\oplus\widetilde{B}_j, \widetilde{B}_1\oplus\widetilde{B}_2\oplus\widetilde{B}_3$) are of the form $E(x,r_1,r_2,r_3)$ as in Eq. (A.11) our analysis suffices to prove that $\widetilde{B}_1\widetilde{B}_2\widetilde{B}_3$ is a hard-core function for f.

□

A.2 Proof of Theorem 4.5.7

Lemma A.2.1 (two-regular)**.** *If $\mathcal{G}$ is a family of bijective functions, then $\mathcal{F}$ is a family of two-regular functions.*

Proof. For every $y\in \mathrm{Im}\, f_{k'}\subseteq R$, where $k'=(k_1,k_2)$:

1. Since $\mathrm{Im}\, f_{k'}=\mathrm{Im}\, g_{k_1}$ and g_{k_1} is bijective, there exist unique $x_1 := g_{k_1}^{-1}(y)$ such that $f_{k'}(x,0)=g_{k_1}(x)=y$.
2. Since $\mathrm{Im}\, f_{k'}=\mathrm{Im}\, g_{k_2}$ and g_{k_2} is bijective, there exist unique $x_2 := g_{k_2}^{-1}(y)$ such that $f_{k'}(x,1)=g_{k_2}(x)=y$.

Therefore, we conclude that:

$$\forall\, y\ \in \mathrm{Im}\, f_{k'}:\ f_{k'}^{-1}(y) := \{(g_{k_1}^{-1}(y),0),(g_{k_2}^{-1}(y),1)\} \qquad (A.21)$$

□

Lemma A.2.2 (trapdoor)**.** *If $\mathcal{G}$ is a family of bijective trapdoor functions, then $\mathcal{F}$ is a family of trapdoor functions.*

Proof. Let $y \in \operatorname{Im} f_{k'} \subseteq R$. We construct the following inversion algorithm:

$Inv_{\mathcal{F}}(k', y, t'_k)$

1 : **/** $t'_k = (t_{k_1}, t_{k_2}), k' = (k_1, k_2)$
2 : $x_1 := Inv_{\mathcal{G}}(k_1, y, t_{k_1})$
3 : $x_2 := Inv_{\mathcal{G}}(k_2, y, t_{k_2})$
4 : **return** $(x_1, 0)$ and $(x_2, 1)$

□

Lemma A.2.3 (one-way)**.** *If $\mathcal{G}$ is a family of bijective, one-way functions, then $\mathcal{F}$ is a family of one-way functions.*

Proof. We prove it by contradiction. We assume that a QPT adversary $\mathcal{A}$ can invert a function in $\mathcal{F}$ with non-negligible probability P (i.e. given $y \in \operatorname{Im} f_{k'}$ to return a correct preimage of the form (x', b) with probability P). We then construct a QPT adversary $\mathcal{A}'$ that inverts any function in $\mathcal{G}$ with the same non-negligible probability P reaching the contradiction since $\mathcal{G}$ is one-way by assumption.

From Eq. (A.21) we know the two preimages of y are: (i) $(g_{k_1}^{-1}(y), 0)$ and (ii) $(g_{k_2}^{-1}(y), 1)$. We now construct an adversary $\mathcal{A}'$ that for $g_k : D \to R$, inverts any image $y = g_k(x)$ with the probability $P/2$.

$\mathcal{A}'(k_c, y)$

1 : $r \leftarrow_{\$} \{0,1\}$
2 : $k_r := k_c$
3 : $(k_{r'}, t_{k_{r'}}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$
4 : **if** $r == 0$ **then**
5 : $\quad k' := (k_r, k_{r'})$
6 : **else**
7 : $\quad k' := (k_{r'}, k_r)$
8 : $(x', b) \leftarrow \mathcal{A}(k', y)$
9 : **if** $((b == r) \wedge (g_{k_r}(x') == y)$ **then**
10 : $\quad$ **/** $\mathcal{A}$ returns correct preimage that also corresponds to the challenge of $\mathcal{A}'$
11 : $\quad$ **return** x'
12 : **else /** $\mathcal{A}$ failed in giving any of the preimages (happens with probability $1 - P$)
13 : $\quad$ **/** or the preimage returned corresponds to the r' that is not the challenge (happens with probability $P/2$)
14 : $\quad$ **return** 0

The inversion algorithm succeeds with $1-((1-P)+P/2) = P/2$ and thus reaches a contradiction. □

Lemma A.2.4 (second preimage resistance)**.** *If $\mathcal{G}$ is a family of bijective, one-way functions, then, any function $f \in \mathcal{F}$ is second preimage resistant.*

Proof. Assume there exists a PPT adversary $\mathcal{B}$ that given $k' = (k_1, k_2)$ and $(y,(x,b))$ such that $f_{k'}(x,b) = y$ can find (x',b') such that $f_{k'}(x',b') = y$ with non-negligible probability P. From Eq. (A.21) we know that the two preimages have different b's. We now construct a PPT adversary $\mathcal{B}'$ that inverts the function g_{k_c} with the same probability P, reaching a contradiction:

$\mathcal{B}'(k_c, y)$

---

1 : $(k_2, t_{k_2}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$

2 : $x_2 \leftarrow g_{k_2}^{-1}(y)$ **/** using the trapdoor t_{k_2}

3 : $k' := (k_c, k_2)$

4 : $(x,0) \leftarrow \mathcal{B}(k', y, (x_2, 1))$ **/** where y is an element from the image of $f_{k'}$

5 : **if** $f_{k'}(x,0) == f_{k'}(x_2,1) == y$

6 : **return** x

7 : **else /** $\mathcal{B}$ failed to find a second preimage; happens with probability $(1-P)$

8 : **return** 0

□

Lemma A.2.5 (quantum-safe)**.** *If $\mathcal{G}$ is a family of quantum-safe trapdoor functions, with properties as above, then $\mathcal{F}$ is also a family of quantum-safe trapdoor functions.*

Proof. The properties that require to be quantum-safe is the one-wayness and second preimage resistance. Both these properties of $\mathcal{F}$ that we derived above were proved using reduction to the hardness (one-wayness) of $\mathcal{G}$. Therefore if $\mathcal{G}$ is quantum-safe, its one-wayness is also quantum-safe and thus both properties of $\mathcal{F}$ are also quantum-safe. □

A.3 Proof of Theorem 4.5.11

In the following, we will denote by $f(s,e,c)$, the function $\texttt{REG2.Eval}_{\mathsf{P}}(k,(s,e,c))$ for k the index function obtained by $\texttt{REG2.Gen}_{\mathsf{P}}(1^n)$, and by s_0, e_0 the trapdoor information associated with this function f.

We now prove separately the δ-2 regularity, collision resistance, one-wayness and trapdoor property of the function in Definition 4.5.10.

A.3.1 δ-2 regularity

Here we describe how to achieve δ-2 regularity using the construction `FromInj` and specifically, the function in Definition 4.5.10.

This reduces to ensuring that the two function inputs (s,e) and $(s-s_0,e-e_0)$ both lie within the domain of the function. The input (s,e) is the result of the inversion algorithm, so it is by definition inside the domain. Additionally, as the first element of the domain is only required to be in $\mathbb{Z}_q^n$ and as $\mathbb{Z}_q$ is closed with subtraction $\bmod q$, then $s-s_0 \in \mathbb{Z}_q^n$ for any s, $s_0 \in \mathbb{Z}_q^n$. On the other hand, the second element of the domain is required to be in $\mathbb{Z}^m$, such that each component is bounded in absolute value by some value μ. In this case, we are not guaranteed that adding or subtracting two such elements the result is still in the domain. What we want to ensure is that with (at least) constant probability over the choice of (s,e) and (s_0,e_0), the result $(s-s_0,e-e_0)$ is in the domain of the function.

It is not difficult to show that if (s_0,e_0) is chosen arbitrarily from the domain of the function, then $(s-s_0,e-e_0)$ lies within the domain of the function only with inverse exponential in m probability. This is why we consider restricting e_0 to be within a subset of the domain. By suitable choice of this subset we can make the success probability (of having two preimages) – seen as a function in m – to be at least a constant value. Firstly, we remark that the exact probability of success can be explicitly computed. Indeed, if the trapdoor noise e_0 is sampled from a Gaussian of dimension m, and standard deviation σ, and if the noise e_1 is sampled uniformly from an hypercube C of length 2μ (both distribution being centered on 0) then the probability that e_0+e_1 is still inside C is:

$$\left(\operatorname{erf}\left(\frac{\sqrt{2}\mu}{\sigma}\right)-\frac{\sigma}{\sqrt{2\pi}\mu}\left(1-\exp\left(-2\left(\frac{\mu}{\sigma}\right)^2\right)\right)\right)^m$$

However, for simplicity, and because we do not aim to find optimal parameters, we will use a (simpler) lower bound of this probability (that will be less efficient by a factor of $\sqrt{m}$). To do that, remark that using Lemma 2.5 in [Reg05], we have that if $e_0 \in \mathbb{Z}^m$, such that each component of e_0 is sampled from a Gaussian distribution with parameter $\alpha' q$, then we have that every component of the vector e_0 is less than $\mu' := \alpha' q\sqrt{m}$ with overwhelming probability as m increases. So one can remark that,

up to a negligible term, the Gaussian distribution with parameter $\alpha' q$ is "closer to 0" than the uniform distribution on $[-\alpha' q\sqrt{m}; \alpha' q\sqrt{m}]$ for sufficiently large m (i.e. for any x, the integral between $-x$ and x of the Gaussian distribution is bigger, up to a negligible term, than the integral of the uniform distribution). Therefore, to obtain a lower bound on the probability of having two preimages, we can consider that e_0 is sampled according to the uniform distribution on a hypercube of length $2\alpha' q\sqrt{m}$ rather than according to the Gaussian distribution of parameter $\alpha' q$. This simplifies our analysis, and allows us to find the subset in which e_0 must reside, as seen in the following lemma. Note also that if one does not want to do any assumption on the input distribution, and only assume that the infinity norm is smaller than μ', then the same Lemma applies with the constant 4 replaced by 2.

Lemma A.3.1 (Domain Addition)**.** *Let $V = \mathbb{R}^m$ be a vector space of dimension m, and let $\mathcal{D}_{m,\mu}$ be the uniform distribution inside the hypercube of dimension m and length 2μ centered on 0. Then, for any $\mu' < \mu$, we have:*

$$P_{m,\mu,\mu'} := \Pr\left[||e_0 + e_1||_\infty \leq \mu \mid e_0 \leftarrow \mathcal{D}_{m,\mu'}, e_1 \leftarrow \mathcal{D}_{m,\mu}\right] = \left(1 - \frac{\mu'}{4\mu}\right)^m$$

Moreover, if $\mu' = O(\frac{\mu}{m})$ then the probability $P_{m,\mu,\mu'}$ becomes lower bounded by a positive constant.

Proof. As $||e_0 + e_1||_\infty$ must be less than μ, which means that each component of the sum vector must be less than μ, and as each component of the 2 vectors e_0 and e_1 was independently sampled, then we can simplify our proof by considering that e_0 and e_1 are vectors in $\mathbb{R}$, essentially determining $P_{1,\mu,\mu'}$ and then, we can compute $P_{m,\mu,\mu'} = {P_{1,\mu,\mu'}}^m$.

Then, let us denote by E_1 the random variable sampled uniformly from $[-\mu,\mu]$, E_0 the random variable sampled uniformly from $[-\mu',\mu']$ and E the random variable obtained as $E = E_1 + E_0$. Therefore, $P_{1,\mu,\mu'} = Pr[-\mu \leq E \leq \mu]$.

Now, we can compute the density function of E using convolution:

$$f_E(e) = \int_{-\infty}^{\infty} f_{E_1}(e_1) \cdot f_{E_0}(e - e_1)\, de_1$$

where f_{E_1} and f_{E_0} are the probability density functions of E_1 and E_0 ($f_{E_1}(e_1) = \frac{1}{2\mu}$, when $e_1 \in [-\mu,\mu]$ and 0 elsewhere and $f_{E_0}(e_0) = \frac{1}{2\mu'}$, when $e_0 \in [-\mu',\mu']$ and 0 elsewhere).

Then, we are only interested in the cases when both the values of $f_{E_1}(e_1)$ and $f_{E_0}(e -$

e_1) are non-zero and for this we need to consider 3 cases for e, given by the intervals: $e \in [-\mu-\mu', \mu'-\mu] \bigcup [\mu'-\mu, \mu-\mu'] \bigcup [\mu-\mu', \mu+\mu']$. Thus, we can derive:

$$f_E(e) = \begin{cases} \int_{-\mu}^{e+\mu'} \frac{1}{4\mu\mu'} de_1 \ , e \in [-\mu-\mu', \mu'-\mu] \\ \int_{e-\mu'}^{e+\mu'} \frac{1}{4\mu\mu'} de_1 \ , e \in [\mu'-\mu, \mu-\mu'] \\ \int_{e-\mu'}^{\mu} \frac{1}{4\mu\mu'} de_1 \ , e \in [\mu-\mu', \mu+\mu'] \end{cases}$$

Finally, we have that $Pr[-\mu \leq E \leq \mu] = \int_{-\mu}^{\mu} f_E(e)\, de = \int_{-\mu}^{-\mu+\mu'} f_E(e)\, de + \int_{-\mu+\mu'}^{\mu-\mu'} f_E(e)\, de + \int_{\mu-\mu'}^{\mu} f_E(e)\, de = \int_{-\mu}^{-\mu+\mu'} \frac{e+\mu'+\mu}{4\mu\mu'} de + \int_{-\mu+\mu'}^{\mu-\mu'} \frac{1}{2\mu} de + \int_{\mu-\mu'}^{\mu} \frac{\mu+\mu'-e}{4\mu\mu'} de = 1 - \frac{\mu'}{4\mu}$.

Consequently, we have $P_{m,\mu,\mu'} = (1 - \frac{\mu'}{4\mu})^m$.

Now, given that μ is a function of m, $\mu = \mu(m)$, we want to determine the values of μ', such that this probability (seen as a function in m) is at least a positive constant number.

- If $\lim_{m\to\infty} \frac{\mu'}{\mu} = 0$, then:

$$\lim_{m\to\infty} \left(1 - \frac{\mu'}{4\mu}\right)^m = \lim_{m\to\infty} \left(1 - \frac{\mu'}{4\mu}\right)^{\frac{4\mu}{\mu'}\frac{\mu' m}{4\mu}} = \left(\frac{1}{e}\right)^{\lim_{m\to\infty} \frac{\mu' m}{4\mu}}$$

  Now, what we require is that $\lim_{m\to\infty} \frac{\mu' m}{4\mu} = c \geq 0$, where c is a constant, as then, we have that the probability of success is at least a constant $\geq \left(\frac{1}{e}\right)^c$.

- If $\lim_{m\to\infty} \frac{\mu'}{\mu} > 0$ (and less than 1, as $0 < \mu' < \mu$), then:

$$\lim_{m\to\infty} \left(1 - \frac{\mu'}{4\mu}\right)^m = 0$$

Consequently, it is clear that in order to get a positive constant lower bound for the success probability, we must have:

$$\mu' = c \cdot \frac{4\mu}{m}, \ c \geq 0$$

Thus, in our case, if e_1 is sampled uniformly on a hypercube of length 2μ and e_0 from a Gaussian with parameter $\alpha' q$, by replacing the actual values of $\mu = \alpha q\sqrt{m}$ and $\mu' := \alpha' q\sqrt{m}$, what we require is that:

$$\alpha' = c \cdot \frac{4\alpha}{m}, \ c \geq 0$$

□

A.3.2 Collision resistance

We start by the observation that for the choices of Definition 4.5.10, no QPT adversary can infer the trapdoor information (s_0, e_0), as determining s_0 from $k = (A, b_0)$ would be equivalent to solving $\mathrm{LWE}_{q,\bar{\Psi}_{\alpha' q}}$:

Corollary A.3.2 (One-wayness of the trapdoor [Reg05, Theorem 1.1])**.** *Under the* SIVP_γ *(with* $\gamma = \mathrm{poly}(n)$*) assumption, no QPT adversary can recover the trapdoor information* (s_0, e_0).

Lemma A.3.3 (Collision resistance)**.** *The function* f *defined in Definition 4.5.10 is collision resistant if the parameters are chosen accordingly to Theorem 4.5.11 assuming that* SIVP_γ *is hard.*

Proof. By contradiction, let us suppose that this function is not collision resistant. Then there exist two pairs (s_1, e_1), (s_2, e_2) such that $f(s_1, e_1, 0) = y = f(s_2, e_2, 1)$. Note that the last bits are necessary different since the two functions that fix the last bit, are injective when the error is smaller than r_{max} (according to [MP12, Theorem 5.4]). By the definition of f, $||e_1||_\infty \leq \mu$ and $||e_2||_\infty \leq \mu$, i.e. both e_1 and e_2 has Euclidean norm smaller than $\sqrt{m}\mu$. Then, by definition, $y = f(s_2, e_2, 1) = f(s_2, e_2, 0) + f(s_0, e_0, 0) = A(s_2 + s_0) + (e_2 + e_0)$. Now, we remark that with overwhelming probability (over the choice of the trapdoor), $||e_0||_2 \leq \mu'\sqrt{m}$ as stated in [Reg05, Lemma 2.5], so in this case, $||e_2 + e_0||_2 \leq \sqrt{m}(\mu + \mu') \leq r_{max}$ (last assumption of Theorem 4.5.11). Then, according to [MP12, Theorem 5.4], there is exactly one element (s, e) with e of length smaller than r_{max} such that $As + e = y$. Because (s_1, e_1) is a solution, we then have that: $s_2 + s_0 = s_1$ and $e_2 + e_0 = e_1$, i.e. $e_0 = e_1 - e_2$ and $s_0 = s_1 - s_2 \mod q$. Hence, it is possible to deduce the trapdoor information s_0 and e_0 from the collision pair, which is impossible by Corollary A.3.2. □

A.3.3 One-wayness

One could imagine that the one-wayness of the resulting function of Definition 4.5.10 is implied by the one-wayness of the function in [MP12] (as is the case in Lemma 4.5.4). However, we need more care here, since in our construction the error term e is not sampled from a Gaussian distribution with suitable parameters (unlike the error term e_0). [1]

[1] While other ways to prove the one-wayness are possible, we give here one proof that uses the previous two lemmata.

Lemma A.3.4 (Collision resistance to one-wayness)**.** *Let $f : A \to B$, where A is finite and can be efficiently sampled uniformly and let C be the set of all $y \in B$ that admit 2 preimages. If f is a collision resistant function that admits with non-negligible probability two preimages for any y from its image and if $\frac{|f^{-1}(C)|}{|A|}$ is non-negligible, then f restricted to the set $f^{-1}(C)$ is a one-way function.*

Proof. By contradiction: suppose that f is not one-way on C, i.e. with a non-negligible probability we can find a preimage of y for y uniformly sampled in C, and from this we can show how to find a collision. The idea is to sample an input $x \in A$, and then compute $y := f(x)$. Then, as $\frac{|f^{-1}(C)|}{|A|}$ is non-negligible, we know that with non-negligible probability this y will have two preimages. Now, with non-negligible probability, this function will be easy to invert and one gets x'. Because we sample uniformly at the step before, we have the same probability to sample one image or the other, so with probability $1/2$, $x' \neq x$, therefore, we found a collision. □

Corollary A.3.5 (One-wayness from Lemma A.3.3 and Lemma A.3.4)**.** *The function defined in Definition 4.5.10 is one-way for all y that admit two preimages, under the* SIVP_γ *hardness assumption, when the parameters are chosen accordingly to Theorem 4.5.11.*

A.3.4 Trapdoor

We want to prove that using the trapdoor information of the `REG2` construction, which consists of (s_0, e_0) and t_k, the trapdoor information of the `LWE` function, we can efficiently derive the preimages of an output b of `REG2.Eval`. Firstly, we notice that to find all the preimages, we can simply run `LWE.Inv` on b as well as on $b - b_0$ and if we succeed we take only the preimages that lie in the input domain, i.e. whose error part e is bounded in infinity norm by μ: $||e||_\infty \leq \mu$. Because the function is injective, these are all the possible preimages. However, because we are interested only in the case when there are exactly two preimages, the function `REG2.Inv` can also do the following: we first run `LWE.Inv` on b and obtain (s_1, e_1). Then, the inversion is completed by returning $(s_1, e_1, 0)$ and $(s_1 - s_0, e_1 - e_0, 1)$, which are both valid preimages, if and only if the function has two preimages (see Lemma A.3.3 for more details).

A.4 Proof of Lemma 4.5.12

Proof. Using the following explicit values for the parameters of the Micciancio and Peikert injective trapdoor function [MP12], we want to prove that they fulfil all of the requirements of Theorem 4.5.11:

$$
\begin{aligned}
n &= \lambda \\
k &= 5\lceil \log(n) \rceil + 21 \\
q &= 2^k \\
\bar{m} &= 2n \\
\omega &= nk \\
m &= \bar{m} + \omega \\
\mu &= \left\lceil 2mn\sqrt{2+k} \right\rceil \\
\mu' &= \mu/m \\
B &= 2
\end{aligned}
$$

and α, α', C are defined as in Theorem 4.5.11. Now, let us prove that these parameters satisfy all the requirements.

- The first three requirements are trivially satisfied.

- In the forth condition, the only difficulty is to show that $\alpha < 1$. By definition,

$$
\begin{aligned}
\alpha = \frac{m\mu}{\sqrt{m}mq} &= \frac{\mu}{\sqrt{m}q} = \frac{\lceil 2mn\sqrt{2+k} \rceil}{\sqrt{m}q} \leq \frac{4mn\sqrt{2+k}}{\sqrt{m}q} \leq \frac{8mn\sqrt{k}}{\sqrt{m}q} \\
&\leq \frac{8\sqrt{m}nk}{q} \leq \frac{8\sqrt{2n+nk}nk}{2^{21}n^5} \leq \frac{8\sqrt{2nk}nk}{2^{21}n^5} \leq \frac{16(nk)^{3/2}}{2^{21}n^5} \\
&\leq \frac{16(n(5(\log(n)+1)+21))^{3/2}}{2^{21}n^5} \leq \frac{16(5\times 21n^2)^{3/2}}{2^{21}n^5} \leq \frac{16\times 1076n^3}{2^{21}n^5} < \frac{1}{n^2} \leq 1
\end{aligned}
$$

- Now, let us show the fifth condition, i.e. $\alpha' q \geq 2\sqrt{n}$. First we note that $\alpha' q := \frac{\mu}{\sqrt{m}m} \geq 2\sqrt{n} \Leftrightarrow \mu \geq 2\sqrt{n}m\sqrt{m} = 2mn\sqrt{2+k}$. Then, by defining $\mu = \lceil 2mn\sqrt{2+k} \rceil$, the condition is satisfied.

- For the fifth condition, i.e. $\frac{n}{\alpha'}$ is $\mathsf{poly}(n)$, we just need to remark that $1/\alpha' = \frac{m^{3/2}q}{\mu} < m^{3/2}q$, and that both m and q are $\mathsf{poly}(n)$.

- Finally, to show that the last condition is satisfied, we note that:

$$\sqrt{m}\mu < \frac{q}{2B\sqrt{\left(C\cdot(\alpha\cdot q)\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1}} - \mu'\sqrt{m} \tag{A.22}$$

$$= \frac{q}{4\sqrt{\left(C\cdot\frac{\mu}{\sqrt{m}}\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1}} - \frac{\mu}{\sqrt{m}} \tag{A.23}$$

if and only if:

$$A := 4\left(\sqrt{m}+\frac{1}{\sqrt{m}}\right)\mu\sqrt{\left(C\cdot\frac{\mu}{\sqrt{m}}\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1} \leq q$$

Now, let us suppose that $k := u\lceil\log(n)\rceil + v$ with $u \leq 5$ and $v \geq 19$ and we need to find u, v such that $A \leq 2^k$. Note that we will include v in some constants and then find the good v at the end. First, remark that:

$$\sqrt{m}+\frac{1}{\sqrt{m}} = \sqrt{m}\left(1+\frac{1}{m}\right) = \sqrt{m}\left(1+\frac{1}{n(2+k)}\right) \tag{A.24}$$

$$\leq \sqrt{m}\left(1+\frac{1}{2+k}\right) \leq \sqrt{m}\underbrace{\left(1+\frac{1}{2+v}\right)}_{\gamma_0} = \gamma_0\sqrt{m} \tag{A.25}$$

So now, we have:

$$
\begin{aligned}
A &\leq 4C\gamma_0\mu^2\sqrt{kn}\sqrt{\left(1+\sqrt{\frac{2}{k}}+\frac{1}{\sqrt{k}}\right)^2+\frac{1}{kn\left(C\cdot\frac{\mu}{\sqrt{m}}\right)^2}} \\
&= 4C\gamma_0\left\lceil 2mn\sqrt{2+k}\right\rceil^2\sqrt{kn}\sqrt{\left(1+\sqrt{\frac{2}{k}}+\frac{1}{\sqrt{k}}\right)^2+\frac{1}{kn\left(C\cdot\frac{\lceil 2mn\sqrt{2+k}\rceil}{\sqrt{m}}\right)^2}} \\
&\leq 4C\gamma_0\left\lceil 2mn\sqrt{2+k}\right\rceil^2\sqrt{kn}\underbrace{\sqrt{\left(1+\sqrt{\frac{2}{v}}+\frac{1}{\sqrt{v}}\right)^2+\frac{1}{v\left(2C\sqrt{2+v}\right)^2}}}_{\gamma_1} \\
&\leq 4C\gamma_0\gamma_1\left(2mn\sqrt{2+k}+1\right)^2\sqrt{kn} = 4C\gamma_0\gamma_1\left(2n^2(2+k)^{3/2}+1\right)^2\sqrt{kn} \\
&= 16C\gamma_0\gamma_1 n^4(2+k)^3\left(1+\frac{1}{2n^2(2+k)^{3/2}}\right)^2\sqrt{kn} \\
&\leq 16C\gamma_0\gamma_1 n^4(2+k)^3\underbrace{\left(1+\frac{1}{2(2+v)^{3/2}}\right)^2}_{\gamma_2}\sqrt{kn} \\
&\leq 16C\gamma_0\gamma_1\gamma_2 n^4\left(k\left(1+\frac{2}{k}\right)\right)^3\sqrt{kn} \leq 16C\gamma_0\gamma_1\gamma_2 n^4k^3\underbrace{\left(1+\frac{2}{v}\right)^3}_{\gamma_3}\sqrt{kn} \\
&\leq 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2}\left(u\lceil\log(n)\rceil+v\right)^{7/2} = 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2}v^{7/2}\left(1+\frac{u\lceil\log(n)\rceil}{v}\right)^{7/2} \\
&\leq 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2}v^{7/2}\left(1+\frac{5\lceil\log(n)\rceil}{19}\right)^{7/2} \leq 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2}v^{7/2}3n^{1/2} \\
&\leq 48C\gamma_0\gamma_1\gamma_2\gamma_3 v^{7/2}n^5
\end{aligned}
$$

Finally, we observe that if $v=21$ and $u=5$, we have $A\leq 2^{v+u\lceil\log(n)\rceil}=2^k$, which concludes the proof.

□

Appendix B

App: Malicious QFactory

B.1 Function Construction proofs

B.1.1 Proof of Lemma 5.3.2

To prove that d_0 is a hardcore predicate of k, we must prove that for any QPT adversary $\mathcal{A}$, we have:

$$\Pr_{\substack{s_0 \leftarrow\$ \mathbb{Z}_q^n \\ e_0 \leftarrow\$ E^m \\ d_0 \leftarrow\$ \{0,1\}}} [\mathcal{A}(1^n, K, g_K(s_0, e_0, d_0)) = d_0(t_k)] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{B.1}$$

Using the definitions of the 2 functions, we can express it as:

$$\Pr_{\substack{s_0 \leftarrow\$ \mathbb{Z}_q^n \\ e_0 \leftarrow\$ E^m \\ d_0 \leftarrow\$ \{0,1\}}} [\mathcal{A}(1^n, K, Ks_0 + e_0 + d_0 \cdot v) = d_0] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{B.2}$$

This is equivalent to proving that the distributions $D_1 = \{K, Ks_0 + e_0\}$ and $D_2 = \{K, Ks_0 + e_0 + v\}$ are indistinguishable to any QPT adversary [1]. Equivalently, by considering each of the m elements of the vectors from the 2 distributions, we need to show that:

$$\{K_i, \langle K_i, s_0 \rangle + e_{0,i}\}_{i=1}^m \overset{c}{\approx} \{K_i, \langle K_i, s_0 \rangle + e_{0,i} + v_i\}_{i=1}^m \tag{B.3}$$

where K_i is the i-th row of K and $e_{0,i}$ is the i-th element of the vector e_0.
Using the decisional LWE assumption we know that for $\{u_i\}_{i=1}^m$ uniformly sampled from $\mathbb{Z}_q$, we have [2]:

$$\{K_i, \langle K_i, s_0 \rangle + e_{0,i}\}_{i=1}^m \overset{c}{\approx} \{K_i, u_i\}_{i=1}^m \tag{B.4}$$

[1] It is also easy to write an explicit reduction

[2] this holds because the function parameters given in Lemma 4.5.12 are chosen to make $Ks_0 + e_0$ indistinguishable from a random vector by a direct reduction to LWE

Then, as v is a fixed constant vector, we also have that:

$$\{K_i, \langle K_i, s_0\rangle + e_{0,i} + v_i\}_{i=1}^m \overset{c}{\approx} \{K_i, u_i\}_{i=1}^m \overset{c}{\approx} \{K_i, \langle K_i, s_0\rangle + e_{0,i}\}_{i=1}^m \tag{B.5}$$

which completes the proof.

B.1.2 Proof of Theorem 5.3.4, homomorphicity

To prove that g_K is homomorphic, we notice that:

$$\begin{aligned} g_K(s_1,e_1,d_1) + g_K(s_2,e_2,d_2) &= \bar{g}_K(s_1,e_1) + d_1 \cdot v + \bar{g}_K(s_2,e_2) + d_2 \cdot v \bmod q \\ &= \bar{g}_K(s_1 + s_2 \bmod q, e_1 + e_2) + (d_1 + d_2) \cdot v \bmod q \\ &= g_K(s_1 + s_2 \bmod q, e_1 + e_2, d_1 \oplus d_2) \end{aligned}$$

where for the last equality we used the fact that if $d_1, d_2 \in \{0,1\}$, then $d_1 \cdot \frac{q}{2} + d_2 \cdot \frac{q}{2} \bmod q = (d_1 \oplus d_2) \cdot \frac{q}{2} \bmod q$.

We make the following remark: the proof is constructed for the case when $\bar{g}$ is perfectly homomorphic, but it also holds in the case when $\bar{g}$ is homomorphic with high probability, resulting in g being homomorphic with the same high probability.

Note also that we can easily notice the homomorphicity property of the defined function h:

$$\begin{aligned} h(s_1,e_1,d_1) \oplus h(s_2,e_2,d_2) &= d_1 \oplus d_2 \\ = h(s_1 + s_2 \bmod q, e_1 + e_2 \bmod q, d_1 \oplus d_2) \end{aligned} \tag{B.6}$$

B.1.3 Proof of Theorem 5.3.4, one-wayness

To prove the **one-wayness** of g_K, we are going to reduce it to the one-wayness of $\bar{g}_K$. Thus, we assume there exists a QPT adversary $\mathcal{A}$ that can invert g_K with probability P and we construct a QPT adversary $\mathcal{A}'$ inverting $\bar{g}_K$ with probability $P/2$.
To show this reduction we will use the fact that g_K is injective, implying that for an image y, there exists an unique preimage (s,e,d) such that $g_K(s,e,d) = y$.

$Invert_{\mathcal{A}',K}(y)$

1 : $d \leftarrow_\$ \{0,1\}$

2 : $y' \leftarrow_\$ y + d \cdot v$

3 : **if** $(d == 0)$ **then**

4 : $\quad (s', e', d') \leftarrow \mathcal{A}_K(y)$ // from the injectivity of g_K we know that if $\mathcal{A}$ succeeds then $d' = 0$

5 : $\quad$ // and (s', e') is the preimage of $\bar{g}$

6 : $\quad$ **return** (s', e')

B.2 Probability of guessing two predicates

Lemma B.2.1 (Implication of guessing two predicates)**.**
Let $(a,b) \in \{0,1\}^2$ be two bits sampled uniformly at random. Let f be any function of (a,b) (eventually randomized). Then if $\mathcal{A}$ is an adversary such that $\Pr[\mathcal{A}(f(a,b)) = (a,b)] \geq 1/4 + \frac{1}{\mathsf{poly}(n)}$ *(where the probability is taken over the choice of a and b, the randomness of f and $\mathcal{A}$), then either:*

- *$\mathcal{A}$ is good to guess a, i.e.:*
  $P_1 = \Pr\left[\tilde{a} = a \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq \frac{1}{2} + \frac{1}{poly_1(n)}$
- *Or $\mathcal{A}$ is good to guess b, i.e.:*
  $P_2 = \Pr\left[\tilde{b} = b \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq \frac{1}{2} + \frac{1}{poly_2(n)}$
- *Or $\mathcal{A}$ is good to guess the XOR of a and b, i.e.:*
  $P_\oplus = \Pr\left[\tilde{a} \oplus \tilde{b} = a \oplus b \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq \frac{1}{2} + \frac{1}{poly_3(n)}$

for some polynomials $poly_1, poly_2, poly_3$.

Proof. Let us define the following quantities:

$$\begin{aligned} e_1 &:= \Pr\left[\tilde{a} \neq a \text{ and } \tilde{b} \neq b \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \\ e_2 &:= \Pr\left[\tilde{a} = a \text{ and } \tilde{b} \neq b \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \\ e_3 &:= \Pr\left[\tilde{a} \neq a \text{ and } \tilde{b} = b \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \\ e_4 &:= \Pr\left[\tilde{a} = a \text{ and } \tilde{b} = b \mid (\tilde{a}, \tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \end{aligned} \tag{B.7}$$

From the initial statement we have: $e_4 \geq \frac{1}{4} + \frac{1}{\mathsf{poly}(n)}$.
Now let us assume that $\mathcal{A}$ is in neither of the first 2 cases, meaning that $\mathcal{A}$ is not good at guessing a and not good at guessing b. We will show that in this case $\mathcal{A}$ must certainly be in the third case, i.e. she must be good at guessing $a \oplus b$.

$\mathcal{A}$ is not good at guessing a implies that: $e_2+e_4 \leq \frac{1}{2}+\mathsf{negl}(n)$ and not good at guessing b implies: $e_3+e_4 \leq \mathsf{negl}(n)$. Then we have:

$$\begin{aligned} e_2 &\leq \frac{1}{2}+\mathsf{negl}(n)-e_4 \leq \frac{1}{2}+\mathsf{negl}(n)-\frac{1}{4}-\frac{1}{poly(n)} \\ e_2 &\leq \frac{1}{4}-\frac{1}{poly'(n)} \text{ for some polynomial } poly' \end{aligned} \tag{B.8}$$

And similarly, there exists a polynomial $poly''$ such that:

$$e_3 \leq \frac{1}{4}-\frac{1}{poly''(n)} \tag{B.9}$$

But then the probability that $\mathcal{A}$ guesses correctly the XOR of a and b is defined as e_1+e_4, for which we have:

$$\begin{aligned} e_1+e_4 &= 1-(e_2+e_3) \\ &\geq 1-\left(\frac{1}{2}-\frac{1}{poly'(n)}-\frac{1}{poly''(n)}\right) \geq \frac{1}{2}+\frac{1}{poly_3(n)} \text{ for some polynomial } poly_3 \end{aligned} \tag{B.10}$$

which concludes our proof.

□

B.3 Proof of Malicious-Abort QFactory

Proof of Lemma 5.5.2. The function f_k cannot have more than two preimages by construction, and in the Malicious 4-states QFactory protocol the output y is an image of f_k. So it means that y has exactly one preimage x. So after measuring the last register, the states will be in the state $|0\rangle \otimes |x\rangle \otimes |y\rangle$. Then, after applying U_h, the state becomes $|d\rangle \otimes |x\rangle \otimes |y\rangle$ with $d \in \{0,1\}$. We remark that the first qubit is not entangled with the measured qubits (second and third register) and as a result, the output qubit will be $|d\rangle$, which is indeed in the basis $\{|0\rangle, |1\rangle\}$. □

Proof of Lemma 5.5.3. The analysis of the circuit will be performed only with respect to the basis of the states of the circuit. Let us first examine the first part of the circuit, where we apply $\wedge Z$ between $\left|+_{\frac{\pi}{2}}\right\rangle$ and $|\mathtt{in}_1\rangle = H^{B_1{}^{(1)}} Z^{B_2{}^{(1)}}$ (with $B_1{}^{(1)}$ the basis of $|\mathtt{in}_1\rangle$) and then measure the first qubit in the $|\pm\rangle$ basis. We denote the resulted state by V_1.

The result of this operation is:

- if $B_1{}^{(1)} = 0$, $V_1 = R\left(\pi(B_2^{(1)} + s_{1,1} + 1)\right)\left|+_{\frac{\pi}{2}}\right\rangle \in \left\{\left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\right\}$

- if ${B_1}^{(1)} = 1$, $V_1 = X^{B_2^{(1)}} |0\rangle \in \{|0\rangle, |1\rangle\}$

In other words, the state V_1 belongs to the basis $\mathcal{B}_0 = \left\{ \left|+\frac{\pi}{2}\right\rangle, \left|-\frac{\pi}{2}\right\rangle \right\}$, if ${B_1}^{(1)} = 0$ and to the basis $\mathcal{B}_1 = \{|0\rangle, |1\rangle\}$ if ${B_1}^{(1)} = 1$.

Now, we can think of the circuit as having t states $V_i \in \left\{ |0\rangle, |1\rangle, \left|+\frac{\pi}{2}\right\rangle, \left|-\frac{\pi}{2}\right\rangle \right\}$, where every V_i has the basis ${B_1}^{(i)}$. Then, to compute the output state $|\texttt{out}\rangle$ of $\texttt{Gad}_{\oplus}$, for every $i \in \{1, ..., t\}$ we have to apply CZ between V_i and $|+\rangle$ and then measure the first qubit in the $|\pm\rangle$ basis.

Let us do this step first for V_1. The result is a state $W_1 = X^{s_{1,2}} H V_1$, thus we obtain that: W_1 belongs to the basis $\mathcal{B}_0 = \left\{ \left|+\frac{\pi}{2}\right\rangle, \left|-\frac{\pi}{2}\right\rangle \right\}$, if ${B_1}^{(1)} = 0$ and to the basis $\mathcal{B}_2 = \{|+\rangle, |-\rangle\}$ if ${B_1}^{(1)} = 1$.

Next we do the same operations between V_2 and W_1, the result being a state W_2, then between V_3 and W_2 and so on, therefore, the outcome state is $|\texttt{out}\rangle = W_t$.

We will prove by induction that the state $W_t \in \left\{ |+\rangle, |-\rangle, \left|+\frac{\pi}{2}\right\rangle, \left|-\frac{\pi}{2}\right\rangle \right\}$, where the basis of W_t is given by $B_1 = {B_1}^{(1)} \oplus ... \oplus {B_1}^{(t)}$.

As we have proved already for the basis case $t = 1$, we now prove the induction step. Suppose that $W_n \in \left\{ |+\rangle, |-\rangle, \left|+\frac{\pi}{2}\right\rangle, \left|-\frac{\pi}{2}\right\rangle \right\}$ with basis $B_1 = {B_1}^{(1)} \oplus ... \oplus {B_1}^{(n)}$.

To obtain W_{n+1} we have to apply $\wedge Z$ between V_{n+1} and W_n and then measure the first qubit. Then after computing this, we obtain that the basis of W_{n+1} is B_1 if the basis of V_{n+1} is ${B_1}^{(n+1)} = 0$ and the basis of W_{n+1} is $1 \oplus B_1$ if the basis of V_{n+1} is ${B_1}^{(n+1)} = 1$. In other words, the basis of W_{n+1} is given by $B_1 = {B_1}^{(1)} \oplus ... \oplus {B_1}^{(n)} \oplus {B_1}^{(n+1)}$, which concludes the proof. □

Proof of Lemma 5.5.4. Let us define $\{A_i\}_{i=1}^{t_c}$ as the (binary) random variables whose values are 1 if and only if in the i-th run of Malicious 4-states QFactory the corresponding y_i has two preimages.

In the honest case, all t_c runs of Malicious 4-states QFactory are independents, hence $\{A_i\}_{i=1}^{t_c}$ are also independent.

From the hypothesis, we know that for all i, $\mathbb{E}(A_i) \geq p_a > p_b$. So let us consider $\varepsilon := \mathbb{E}(A_i) - p_b \geq p_a - p_b$.

Then, by using Chernoff-Hoeffding inequality we have:

$$\Pr\left[\frac{1}{t_c} \sum_{i=1}^{t_c} A_i < \mathbb{E}(A_i) - \varepsilon \right] \leq e^{-2\varepsilon^2 t_c} \leq e^{-2(p_a - p_b)^2 t_c}$$

This is equivalent to:

$$\Pr\left[\sum_{i=1}^{t_c} A_i < t_c \cdot p_b \right] < e^{-2(p_a - p_b)^2 t_c} \tag{B.11}$$

As $p_a - p_b$ is constant, we conclude:

$$\Pr\left[\sum_{i=1}^{t_c} A_i \geq t_c \cdot p_b\right] \geq 1 - \mathsf{negl}(t_c) \tag{B.12}$$

□

Proof of Lemma 5.5.5. From Lemma 5.5.4, for a given chunk, the probability to have at least $p_c t_c$ accepted runs is $1 - \mathsf{negl}(t_c)$, i.e. if $t_c = \Omega(n)$, this probability is $\mathsf{negl}(n)$. Then, the probability to accept all n_c chunks is $(1 - \mathsf{negl}(n))^{n_c} = 1 - \mathsf{negl}(n)$ as soon as $n_c = \mathsf{poly}(n)$, which is the case because $t = t_c \times n_c = \mathsf{poly}(n)$. Then, when all the chunks are accepted, the correctness of the output values is assured by Lemma 5.5.3.

□

B.4 Generalisation to pseudo-homomorphic functions

Definition B.4.1 ($(\eta, \mathcal{Z}, \mathcal{Z}_0, \mathcal{D})$-homomorphic family of functions)**.** *Let us consider a family of functions $\{g_k : \mathcal{Z} \to \mathcal{Y} \cup \perp\}_{k \in \mathcal{K}}$, as well as two symmetric binary group relations $*$ and $\star$, with $*$ acting on a set containing $\mathcal{Z}$ and $\mathcal{Z}_0$, $\star$ acting on $\mathcal{Y} \cup \perp$, and so that $\forall y \in \mathcal{Y}, \perp \star y = \perp$. We say that $\{f_k\}_{k \in \mathcal{K}}$ is an $(\eta, \mathcal{Z}, \mathcal{Z}_0, \mathcal{D})$-homomorphic function if $\mathcal{D}$ is a distribution on $\mathcal{Z}_0$ and*

$$\Pr_{\substack{k \leftarrow \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [z * z_0 \in \mathcal{Z} \text{ and } g_k(z) \star g_k(z_0) = g_k(z * z_0) \neq \perp] \geq \eta$$

Note that we do require that z is sampled uniformly from $\mathcal{Z}$, but z_0 is sampled from a distribution $\mathcal{D}$ on $\mathcal{Z}_0$ that may not be uniform.

Definition B.4.2 (δ-2-regular family of functions)**.** *Let us consider a family of functions $\{f_k : \mathcal{X} \to \mathcal{Y} \cup \perp\}_{k \in \mathcal{K}}$. For a fixed k, $\mathcal{Y}^{(2)}$ will be the set of y having two preimages: $\mathcal{Y}_{f_k}^{(2)} = \{y \in \mathcal{Y}, |f_k^{-1}(y)| = 2\}$. Then, this family of functions is said to be δ-2-regular if*

$$\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ x \leftarrow\$ \mathcal{X}}} [f_k(x) \in \mathcal{Y}_{f_k}^{(2)}] \geq \delta$$

Lemma B.4.3 ($(\eta, \mathcal{Z}, \mathcal{Z}_0)$-homomorphicity to δ-2-regularity)**.** *Given a family of functions $\{g_k : \mathcal{Z} \to \mathcal{Y} \cup \perp\}_{k \in \mathcal{K}}$ that is both injective and an $(\eta, \mathcal{Z}, \mathcal{Z}_0)$-homomorphic family of functions, then it's possible to build a family $\{f_{k'} : \mathcal{Z} \times \{0,1\} \to \mathcal{Y} \cup \perp\}_{k' \in \mathcal{K}'}$ that is δ-2-regular, with $\delta = \eta$.*

Proof. Let us consider the following construction. To sample a key $k' \in \mathcal{K}'$, we first sample a key k from $\mathcal{K}$, as well as an $z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0$, and we define $k' = (k, y_0 := f_k(z_0))$. Then, we define $f_{k'}(z,0) = g_k(z)$ and $f_{k'}(z,1) = g_k(z) \star y_0$, also denoted later as $f_{k'}(z,c) = g_k(z) \star (c \cdot y_0)$ for simplicity. Now, we remark that:

$$\Pr_{\substack{k' \leftarrow\$ \mathcal{K}' \\ x \leftarrow\$ \mathcal{X}}}[f_{k'}(x) \in \mathcal{Y}_{f'_k}^{(2)}] = \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z} \\ c \leftarrow\$ \{0,1\}}}[g_k(z) \star (c \cdot g_k(z_0)) \in \mathcal{Y}_{f'_k}^{(2)}] \tag{B.13}$$

$$= \frac{1}{2} \cdot \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \in \mathcal{Y}_{f'_k}^{(2)}] + \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)}] \Big) \tag{B.14}$$

$$\geq \frac{1}{2} \cdot \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \in \mathcal{Y}_{f'_k}^{(2)} \text{ and } \underbrace{z * z_0^{-1} \in \mathcal{Z} \text{ and } g_k(z) = g_k(z * z_0^{-1}) \star g_k(z_0) \neq \perp}_{C_1}] \tag{B.15}$$

$$+ \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)} \text{ and } \underbrace{z * z_0 \in \mathcal{Z} \text{ and } g_k(z) \star g_k(z_0) = g_k(z * z_0) \neq \perp}_{C_2}] \Big) \tag{B.16}$$

$$= \frac{1}{2} \cdot \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \in \mathcal{Y}_{f'_k}^{(2)} | C_1] \cdot \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_1] + \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)} | C_2] \cdot \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_2] \Big) \tag{B.17}$$

Now, we remark that when $z_0 * z \in \mathcal{D}$ and $g_k(z_0) \star g_k(z) = g_k(z_0 * z) \neq \perp$, then $y := g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)}$. More specifically:

- $y \in \mathcal{Y}$ because $g_k(z) \star g_k(z_0) \neq \perp$ and the $\star$ operator is defined on $\mathcal{Y} \cup \perp$
- there are at least two preimages mapping to y, because $y = f_k(z,1) = g_k(z) \star g_k(z_0) = g_k(z * z_0) = f_k(z * z_0, 0)$.
- there are at most two preimages mapping to y: indeed g_k is injective, so both partial functions $f(\cdot,0)$ and $f(\cdot,1)$ are injective, so we cannot have more than two preimages mapping to y.

As a result, we have: $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)} | C_2] = 1$.

Similarly, $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[g_k(z) \in \mathcal{Y}_{f'_k}^{(2)} | C_1] = 1$.

Hence, we can rewrite the above equation as:

$$\Pr_{\substack{k' \leftarrow\$ \mathcal{K}' \\ x \leftarrow\$ \mathcal{X}}}[f_{k'}(x) \in \mathcal{Y}_{f'_k}^{(2)}] \geq \frac{1}{2} \cdot \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_1] + \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_2] \Big) \tag{B.18}$$

Now, as $\{g_k\}_k$ is $(\eta, \mathcal{Z}, \mathcal{Z}_0)$-homomorphic, we have: $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_2] \geq \eta$. By symmetry, we also have: $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_1] \geq \eta$. Indeed:

$$\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[\underbrace{z * z_0^{-1}}_{\hat{z}} \in \mathcal{Z} \text{ and } g_k(z) = g_k(z * z_0^{-1}) \star g_k(z_0) \neq \perp] \tag{B.19}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[\hat{z} \in \mathcal{Z} \text{ and } z \in \mathcal{Z} \text{ and } \hat{z} = z * z_0^{-1} \text{ and } g_k(z) = g_k(\hat{z}) \star g_k(z_0) \neq \perp] \tag{B.20}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ \hat{z} \leftarrow\$ \mathcal{Z}}}[\hat{z} \in \mathcal{Z} \text{ and } z \in \mathcal{Z} \text{ and } \hat{z} = z * z_0^{-1} \text{ and } g_k(z) = g_k(\hat{z}) \star g_k(z_0) \neq \perp] \tag{B.21}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ \hat{z} \leftarrow\$ \mathcal{Z}}}[\hat{z} * z_0 \in \mathcal{Z} \text{ and } g_k(\hat{z} * z_0) = g_k(\hat{z}) \star g_k(z_0) \neq \perp] \tag{B.22}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}}[C_2] \geq \eta \tag{B.23}$$

So $\Pr_{\substack{k' \leftarrow\$ \mathcal{K}' \\ x \leftarrow\$ \mathcal{X}}}[f_{k'}(x) \in \mathcal{Y}_{f'_k}^{(2)}] \geq \eta$, which concludes the proof. □

Appendix C

App: Composable RSP

C.1 Distance Measures for Quantum States

Lemma C.1.1. *For any two density matrices* ρ, σ *it holds that:*

$$\mathrm{Tr}(\rho\sigma) = \frac{1}{2}\left[\mathrm{Tr}\left(\rho^2\right) + \mathrm{Tr}\left(\sigma^2\right)\right] - \frac{1}{2}\left\|\rho - \sigma\right\|_{HS}^2,$$

where the Hilbert-Schmidt norm is defined as: $\|A\|_{HS} = \sqrt{\mathrm{Tr}(A^*A)}$.

Proof. This follows directly from the relation

$$(\rho - \sigma)^2 = \rho^2 - \rho\sigma - \sigma\rho + \sigma^2$$

and the fact that ρ and σ are hermitian. □

The following lemma formalizes the following statement: If $\mathrm{Tr}(\rho\sigma)$ is close to 1, then both ρ and σ must be almost pure, and ρ and σ must be close. Note that Lemma C.1.2 holds in particular for density matrices ρ and σ, despite being stated for a more general class of operators.

Lemma C.1.2. *Let* $\varepsilon \geq 0$ *and* $\mathrm{Tr}\left(\rho\sigma\right) \geq 1 - \varepsilon$ *for two self-adjoint, positive semi-definite operators* ρ, σ *with trace less or equal than 1. Then, it holds that:*

1. $\mathrm{Tr}\left(\rho^2\right) \geq 1 - 2\varepsilon$,
2. $\mathrm{Tr}\left(\sigma^2\right) \geq 1 - 2\varepsilon$,
3. $\left\|\rho - \sigma\right\|_{HS} \leq \sqrt{2\varepsilon}$.

Proof. From Lemma C.1.1, we infer that:

$$\mathrm{Tr}(\rho\sigma) \leq \frac{1}{2}\left[\mathrm{Tr}\left(\rho^2\right) + \mathrm{Tr}\left(\sigma^2\right)\right] \leq \frac{1}{2}\left[\mathrm{Tr}\left(\rho^2\right) + 1\right],$$

by using the non-negativity of Hilbert-Schmidt norm and $\mathrm{Tr}\left(\sigma^2\right) \leq 1$. Hence:

$$\mathrm{Tr}\left(\rho^2\right) \geq 2\,\mathrm{Tr}\left(\rho\sigma\right) - 1 \geq 1 - 2\varepsilon.$$

Similarly, it can be shown that $\mathrm{Tr}\left(\sigma^2\right) \geq 1 - 2\varepsilon$.
Finally, using $\mathrm{Tr}\left(\rho^2\right) \leq 1$ and $\mathrm{Tr}\left(\sigma^2\right) \leq 1$, we obtain:

$$\mathrm{Tr}\left(\rho\sigma\right) \leq 1 - \frac{1}{2}\left\|\rho - \sigma\right\|_{\mathrm{HS}}^2$$
$$\left\|\rho - \sigma\right\|_{\mathrm{HS}}^2 \leq 2\left(1 - \mathrm{Tr}\left(\rho\sigma\right)\right) \leq 2\varepsilon,$$

□

Lemma C.1.3. *Let λ be a security parameter and let ρ, σ be two density matrices. Then, the following statements are equivalent:*

1. $\mathrm{Tr}\left(\rho^2\right) \geq 1 - \mathsf{negl}\left(\lambda\right)$, $\mathrm{Tr}\left(\sigma^2\right) \geq 1 - \mathsf{negl}\left(\lambda\right)$ *and* $\mathsf{TD}\left(\rho - \sigma\right) \leq \mathsf{negl}\left(\lambda\right)$,
2. $\mathrm{Tr}\left(\rho\sigma\right) \geq 1 - \mathsf{negl}\left(\lambda\right)$,

where TD *denotes the trace distance.*

Proof. One direction of the equivalence follows directly from Lemma C.1.2. The other direction follows from the formula in Lemma C.1.1 and the fact that in finite-dimensional spaces the trace norm is equivalent to the Hilbert-Schmidt norm. □

Lemma C.1.4. *Let $\varepsilon_1, \varepsilon_2 \geq 0$. Let further $\mathrm{Tr}\left(\rho_1\rho_2\right) \geq 1 - \varepsilon_1$ and $\mathrm{Tr}\left(\rho_2\rho_3\right) \geq 1 - \varepsilon_2$ for self-adjoint, positive semi-definite operators ρ_1, ρ_2, ρ_3 with trace less than 1. Then it holds that $\mathrm{Tr}\left(\rho_1\rho_3\right) \geq 1 - 3\left(\varepsilon_1 + \varepsilon_2\right)$.*

Proof. From Lemma C.1.2 we know that $\mathrm{Tr}\left(\rho_1^2\right) \geq 1 - 2\varepsilon_1$, $\mathrm{Tr}\left(\rho_3^2\right) \geq 1 - 2\varepsilon_2$, and

$$\left\|\rho_1 - \rho_2\right\|_{\mathrm{HS}} \leq \sqrt{2\varepsilon_1}, \quad \left\|\rho_2 - \rho_3\right\|_{\mathrm{HS}} \leq \sqrt{2\varepsilon_2}.$$

By the triangle inequality for the Hilbert-Schmidt norm, it follows readily that

$$\left\|\rho_1 - \rho_3\right\|_{\mathrm{HS}} \leq \sqrt{2\varepsilon_1} + \sqrt{2\varepsilon_2}$$

and therefore:

$$\left\|\rho_1 - \rho_3\right\|_{\mathrm{HS}}^2 \leq \left(\sqrt{2\varepsilon_1} + \sqrt{2\varepsilon_2}\right)^2 = 2\varepsilon_1 + 2\varepsilon_2 + 4\sqrt{\varepsilon_1}\sqrt{\varepsilon_2} \leq 4\left(\varepsilon_1 + \varepsilon_2\right)$$

where we applied the inequality of the geometric mean to obtain the last bound. Using the formula from Lemma C.1.1, we then conclude that:

$$\begin{aligned}\mathrm{Tr}\left(\rho_1\rho_3\right) &= \frac{1}{2}\left[\mathrm{Tr}\left(\rho_1^2\right) + \mathrm{Tr}\left(\rho_3^2\right)\right] - \frac{1}{2}\left\|\rho_1 - \rho_3\right\|_{\mathrm{HS}}^2 \\ &\geq \frac{1}{2}\left[1 - 2\varepsilon_1 + 1 - 2\varepsilon_2\right] - \frac{1}{2}4\left(\varepsilon_1 + \varepsilon_2\right) \geq 1 - 3\left(\varepsilon_1 + \varepsilon_2\right).\end{aligned}$$

□

Bibliography

[AA11] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 333–342, 2011.

[Aar05] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society of London Series A*, 461:3473–3482, November 2005.

[Aar06] S. Aaronson. Qma/qpoly /spl sube/ pspace/poly: de-merlinizing quantum protocols. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 13 pp.–273, 2006.

[Aar10] Scott Aaronson. Bqp and the polynomial hierarchy. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 141–150, New York, NY, USA, 2010. Association for Computing Machinery.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[ACGK19] Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. Complexity-Theoretic Limitations on Blind Delegated Quantum Computation. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 2019.

[AFK87] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 195–203, New York, NY, USA, 1987. Association for Computing Machinery.

[AS03] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 04, 10 2003.

[BB14] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.*, 560:7–11, 2014.

[BCC+20] Christian Badertscher, Alexandru Cojocaru, Léo Colisson, Elham Kashefi, Dominik Leichtle, Atul Mantri, and Petros Wallden. Security limitations of classical-client delegated quantum computing. *arXiv preprint arXiv:2007.01668*, 2020.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *FOCS*, pages 320–331. IEEE Computer Society, 2018.

[BDG90] José L. Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity II*, volume 22 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1990.

[BDG12] Jose L. Balcazar, Josep Diaz, and Joaquim Gabarro. *Structural Complexity I*. Springer Publishing Company, Incorporated, 2nd edition, 2012.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 517–526, Washington, DC, USA, 2009. IEEE Computer Society.

[BJ15] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low t-gate complexity. In *Annual Cryptology Conference*, pages 609–629. Springer, 2015.

[Bjö16] Andreas Björklund. Below all subsets for some permutational counting problems. In *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 17:1–17:11, 2016.

[BKB+12] Stefanie Barz, Elham Kashefi, Anne Broadbent, Joseph F Fitzsimons, Anton Zeilinger, and Philip Walther. Demonstration of blind quantum computing. *Science*, 335(6066):303–308, 2012.

[BOV+18] Mathieu Bozzio, Adeline Orieux, Luis Trigo Vidarte, Isabelle Zaquine, Iordanis Kerenidis, and Eleni Diamanti. Experimental investigation of practical unforgeable quantum money. *npj Quantum Information*, 4(1):5, 2018.

[Bra18] Zvika Brakerski. Quantum fhe (almost) as secure asclassical. In *Advances in Cryptology – CRYPTO 2018*, pages 67–95, Cham, 2018. Springer International Publishing.

[Bro15a] Anne Broadbent. Delegating private quantum computations. *Canadian Journal of Physics*, 93(9):941–946, 2015.

[Bro15b] Anne Broadbent. How to verify a quantum computation, 2015. Eprint:arXiv:1509.09180.

[CCKM20] Michele Ciampi, Alexandru Cojocaru, Elham Kashefi, and Atul Mantri. Secure quantum two-party computation: Impossibility and constructions. Cryptology ePrint Archive, Report 2020/1286, 2020. `https://eprint.iacr.org/2020/1286`.

[CCKW18] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. Delegated pseudo-secret random qubit generator. *arXiv preprint arXiv:1802.08759*, 2018.

[CCKW19] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. Qfactory: Classically-instructed remote secret qubits preparation. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 615–645. Springer International Publishing, 2019.

[CGK+19] Alexandru Cojocaru, Juan Garay, Aggelos Kiayias, Fang Song, and Petros Wallden. The bitcoin backbone protocol against quantum adversaries. Cryptology ePrint Archive, Report 2019/1150, 2019. `https://eprint.iacr.org/2019/1150`.

[Chi05] Andrew M. Childs. Secure assisted quantum computation. *Quantum Info. Comput.*, 5(6):456–466, September 2005.

[DFPR14] Vedran Dunjko, Joseph F Fitzsimons, Christopher Portmann, and Renato Renner. Composable security of delegated quantum computation. In

International Conference on the Theory and Application of Cryptology and Information Security, pages 406–425. Springer, 2014.

[DK06] Vincent Danos and Elham Kashefi. Determinism in the one-way model. *Physical Review A*, 74(5):052310, 2006.

[DK16] Vedran Dunjko and Elham Kashefi. Blind quantum computing with two almost identical states. *arXiv preprint arXiv:1604.01586*, 2016.

[DKL11] Vedran Dunjko, Elham Kashefi, and Anthony Leverrier. Blind quantum computing with weak coherent pulses. *Physical Review Letters*, 108:200502, 08 2011.

[DNS10] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Secure two-party quantum evaluation of unitaries against specious adversaries. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 685–706, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[DSS16] Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum homomorphic encryption for polynomial-sized circuits. In *Annual Cryptology Conference*, pages 3–32. Springer, 2016.

[Fei86] Joan Feigenbaum. Encrypting problem instances. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 477–488, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[FGK+10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In *International Workshop on Public Key Cryptography*, pages 279–295. Springer, 2010.

[Fit17] Joseph F Fitzsimons. Private quantum computation: an introduction to blind quantum computing and related protocols. *npj Quantum Information*, 3(1):23, 2017.

[FK17] Joseph F Fitzsimons and Elham Kashefi. Unconditionally verifiable blind quantum computation. *Physical Review A*, 96(1):012303, 2017.

[FKD17] Samuele Ferracin, Theodoros Kapourniotis, and Animesh Datta. Towards minimising resources for verification of quantum computations. *arXiv preprint arXiv:1709.10050*, 2017.

[FWJ+14] Heng Fan, Yi-Nan Wang, Li Jing, Jie-Dong Yue, Han-Duo Shi, Yong-Liang Zhang, and Liang-Zhu Mu. Quantum cloning machines and the applications. *Physics Reports*, 544(3):241 – 322, 2014. Quantum cloning machines and the applications.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, page 40–49. IEEE Computer Society, 2013.

[GHZ89] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 25–32, New York, NY, USA, 1989. ACM.

[GMMR13] Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G Rudolph. Efficient universal blind quantum computation. *Physical review letters*, 111(23):230501, 2013.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. *On Yao's XOR-Lemma*, pages 273–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[Gol01] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, Aug 2001.

[GRB+16] Chiara Greganti, Marie-Christine Roehsner, Stefanie Barz, Tomoyuki Morimae, and Philip Walther. Demonstration of measurement-only blind quantum computing. *New Journal of Physics*, 18(1):013020, 2016.

[GRW80] G. C. Ghirardi, Alberto Rimini, and Tullio Weber. A general argument against superluminal transmission through the quantum mechanical measurement process. *Lettere al Nuovo Cimento (1971-1985)*, 27:293–298, 1980.

[GV19] Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1024–1033, 2019.

[Joz05] R. Jozsa. An introduction to measurement based quantum computation. *arXiv: Quantum Physics*, 2005.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[LC18] Ching-Yi Lai and Kai-Min Chung. On statistically-secure quantum homomorphic encryption. *Quantum Info. Comput.*, 18(9–10):785–794, 2018.

[LDT+18] Nana Liu, Tommaso F Demarie, Si-Hui Tan, Leandro Aolita, and Joseph F Fitzsimons. Client-friendly continuous-variable blind and verifiable quantum computing. *arXiv preprint arXiv:1806.09137*, 2018.

[Lia15] Min Liang. Quantum fully homomorphic encryption scheme based on universal quantum circuit. *Quantum Information Processing*, 14(8):2749–2759, 2015.

[Mah18] Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 332–338. IEEE Computer Society, 2018.

[Mau11] Ueli Maurer. Constructive cryptography–a new paradigm for security definitions and proofs. In *Theory of Security and Applications*, pages 33–56. Springer, 2011.

[MDF17] Atul Mantri, Tommaso F Demarie, and Joseph F Fitzsimons. Universality of quantum computation with cluster states and (X, Y)-plane measurements. *Scientific Reports*, 7:42861, 2017.

[MDK15] Tomoyuki Morimae, Vedran Dunjko, and Elham Kashefi. Ground state blind quantum computation on aklt state. *Quantum Info. Comput.*, 15(3-4):200–234, March 2015.

[MDMF17] Atul Mantri, Tommaso F Demarie, Nicolas C Menicucci, and Joseph F Fitzsimons. Flow ambiguity: A path towards classically driven blind quantum computation. *Physical Review X*, 7(3):031004, 2017.

[MF12] Tomoyuki Morimae and Keisuke Fujii. Blind topological measurement-based quantum computation. *Nature communications*, 3:1036, 09 2012.

[MF13] Tomoyuki Morimae and Keisuke Fujii. Blind quantum computation protocol in which alice only makes measurements. *Phys. Rev. A*, 87:050301, May 2013.

[MK13a] Tomoyuki Morimae and Takeshi Koshiba. Composable security of measuring-alice blind quantum computation. *arXiv preprint arXiv:1306.2113*, 2013.

[MK13b] Tomoyuki Morimae and Takeshi Koshiba. Composable security of measuring-alice blind quantum computation. *arXiv preprint arXiv:1306.2113*, 2013.

[MK14] Tomoyuki Morimae and Takeshi Koshiba. Impossibility of perfectly-secure delegated quantum computing for classical client. *arXiv preprint arXiv:1407.1636*, 2014.

[Mor12] Tomoyuki Morimae. Continuous-variable blind quantum computation. *Physical Review Letters*, 109(23):230502, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[MPDF13] Atul Mantri, Carlos A Pérez-Delgado, and Joseph F Fitzsimons. Optimal blind quantum computation. *Physical review letters*, 111(23):230502, 2013.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *In Innovations in Computer Science*. Citeseer, 2011.

[NC00] Michael A Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Nie06] Michael A Nielsen. Cluster-state quantum computation. *Reports on Mathematical Physics*, 57(1):147–161, 2006.

[NS18] Michael Newman and Yaoyun Shi. Limitations on transversal computation through quantum homomorphic encryption. *Quantum Info. Comput.*, 18(11–12):927–948, 2018.

[OTF15] Yingkai Ouyang, Si-Hui Tan, and Joseph Fitzsimons. Quantum homomorphic encryption from quantum codes. *arXiv preprint arXiv:1508.00938*, 2015.

[PCDK11] Anna Pappa, André Chailloux, Eleni Diamanti, and Iordanis Kerenidis. Practical quantum coin flipping. *Physical Review A*, 84(5):052305, 2011.

[PDF15] Carlos A Pérez-Delgado and Joseph F Fitzsimons. Iterated gate teleportation and blind quantum computation. *Physical Review Letters*, 114(22):220502, 2015.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[Pre18] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[RB01] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22):5188, 2001.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[Rys63] Herbert John Ryser. *Combinatorial Mathematics*. Mathematical Association of America, 1963.

[Sho97] 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.

[SIGA05] Valerio Scarani, Sofyan Iblisdir, Nicolas Gisin, and Antonio Acín. Quantum cloning. *Rev. Mod. Phys.*, 77:1225–1256, Nov 2005.

[Sim97] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, 1997.

[SSS09] Louis Salvail, Christian Schaffner, and Miroslava Sotáková. On the power of two-party quantum cryptography. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 70–87, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Sto83] Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 118–126, 1983.

[TKO+16] Si-Hui Tan, Joshua A Kettlewell, Yingkai Ouyang, Lin Chen, and Joseph F Fitzsimons. A quantum approach to homomorphic encryption. *Scientific reports*, 6:33467, 2016.

[Tod91] Seinosuke Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, page 865–877, 1991.

[TOR18] Si-Hui Tan, Yingkai Ouyang, and Peter P. Rohde. Practical somewhat-secure quantum somewhat-homomorphic encryption with coherent states. *Phys. Rev. A*, 97:042308, Apr 2018.

[Vai] Vinod Vaikuntanathan. Advanced topics in cryptography: Lattices. `https://people.csail.mit.edu/vinodv/6876-Fall2015/L13.pdf`.

[VV85] Umesh V. Vazirani and Vijay V. Vazirani. Efficient and secure pseudo-random number generation (extended abstract). In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 193–202, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[WDKA15] Petros Wallden, Vedran Dunjko, Adrian Kent, and Erika Andersson. Quantum digital signatures with quantum-key-distribution components. *Physical Review A*, 91(4):042304, 2015.

[WEH18] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412):303, 2018.

[Yap83] Chee K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287 – 300, 1983.

[YPDF14] Li Yu, Carlos A Pérez-Delgado, and Joseph F Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Physical Review A*, 90(5):050303, 2014.

[Zha20] Jiayu Zhang. Succinct blind quantum computation using a random oracle. *ArXiv*, abs/2004.12621, 2020.