Multi-Party Quantum Cryptography : from Folklore to Real-World

Luka Music

Sorbonne Université - EDITE de Paris

Laboratoire d'Informatique de Sorbonne Université (LIP6) / Quantum Information

Multi-Party Quantum Cryptography: From Folklore to Real-World

Par Luka Music

Thèse de Doctorat d'Informatique

Dirigée par Elham Kashefi et Céline Chevalier

Présentée et soutenue publiquement en Juillet 2021, devant un jury composé de :

- Kashefi Elham, Directrice de Recherche au CNRS, Sorbonne Université, University of Edinburgh, Directrice de thèse
- Chevalier Céline, Maître de Conférences HDR, Université Panthéon Assas Paris 2, Directrice de thèse
- Blazy Olivier, Maître de Conférences HDR, Université de Limoges, Rapporteur
- Alagic Gorjan, Associate Research Scientist, University of Maryland, Rapporteur
- Pointcheval David, Directeur de Recherche au CNRS, ENS Paris
- Mitrokotsa Aikaterini, Professor HDR, University of St. Gallen
- Pappa Anna, Group Leader, Technical University Berlin
- Chailloux André, Chargé de Recherche au CNRS, INRIA

Abstract

The field of quantum cryptography builds upon decades of advances both in classical cryptography and networks. However, contrary to its classical counterparts, it is still in its infancy applicability-wise, even in the scenario where powerful quantum computers are readily available, and more theoretical work is required before it can provide concrete benefits. The first goal is to formalise in rigorous quantum security frameworks the properties of various techniques that have been transposed, often without proper justification, from the classical world. Then, the recent developments in quantum technologies suggest a mostly cloud-based future availability of quantum devices. Therefore, quantum computation and communication cost of protocol participants must be lowered before being useful. Finally, in most situations, additional steps need to be taken to tailor protocols to the specifications of devices. This allows for optimisations both in terms of quantum memory and operation requirements. This thesis contributes to these three aspects by:

1. giving the first general security definition of the Quantum Cut-and-Choose, a technique for proving the correctness of a quantum message;
2. presenting a more realistic framework of security against superposition attacks, where classical protocols run on inherently quantum devices;
3. constructing an efficient delegated multi-party quantum computation protocol, allowing clients to delegate securely to a quantum server a private computation;
4. building a method for verifying the honesty of a quantum server performing computations on behalf of a client with no operation or memory overhead compared to the unprotected computation.

La cryptographie quantique a bénéficié des nombreuses avancées de la cryptographie et théorie des réseaux classiques. Cependant, elle n'en est qu'à ses balbutiement en ce qui concerne son application en condition réelles et approfondir la théorie sous-jacente est un prérequis crucial à l'exploitation de l'intégralité de ses possibilités. Pour cela, il faut tout d'abord formaliser rigoureusement les propriétés de sécurité quantiques des techniques importées de la cryptographie classique, pour l'instant souvent utilisées sans justification. Ensuite, les progrès récents des technologies quantiques tendent à pointer vers un modèle d'accès type client-serveur avec un client faiblement quantique. Dans ce contexte, les protocoles quantiques se doivent d'être les plus frugaux possibles en termes de ressources (mémoire et opération). Enfin, implémenter des protocoles sur des architectures concrètes nécessite de les adapter finement aux machines utilisées afin d'améliorer encore leur optimisation. Cette thèse contribue à ces trois aspects en :

1. proposant une définition du Quantum Cut-and-Choose, technique qui permet de garantir la préparation honnête d'un message quantique ;
2. présentant un cadre de sécurité plus réaliste contre les attaques par superposition, qui garantit la sécurité de protocoles classiques exécutés sur une machine quantique ;
3. construisant un protocole efficace de délégation de calcul multipartite quantique, qui permet à des clients de déléguer un calcul privé à un serveur ;
4. démontrant qu'il est possible de vérifier l'exactitude de calculs quantiques délégués sans aucun impact en terme ressources côté client ou serveur.

Table of Contents

Chapter 1

Introduction

The development of quantum computing and quantum communication builds upon decades of analysis of their classical counterparts. The recent path taken by start-ups and large technological corporations points toward quantum technologies being mostly cloud-accessible. While, as of now, it is possible to access experimental machines through classical networks, several initiatives are working on building quantum networks that would allow remote quantum access to quantum computers.

A wealth of questions arise in this setting from a security perspective, one of which being that clients and servers, reflecting real-life situations, do not necessarily trust each other. Protecting the clients' inputs and computations that are run on dishonest servers is therefore central, and even more so in the quantum domain, as quantum computers are constructed precisely for the purpose of tackling sensitive and/or high added-value computations. Several protocols providing privacy with or without output correctness verifiability have been proposed and proven secure (see e.g. [41], and [52, 58] for a review).

In the context of classical networked machines, Secure Multi-Party Computation (SMPC) is another vital functionality provided by modern cryptography, allowing several players to collaboratively compute a joint function on private input data. The parties are required to exchange messages as no single party possesses the full data required to perform the computation. However, they want to do so while minimising their exposure since they do not trust each other and wish to maintain the privacy of their input (to name some examples: millionaire's problem, coin tossing, voting schemes, etc). The field started with the seminal paper of Yao [135], where two parties want to compute a function of their joint inputs and yet are "Honest-but-Curious" in the sense that they follow the protocol's specifications but wish to extract more information about the other player's input than what is permitted by simply obtaining the final result. This protocol was later made secure against Malicious Adversaries – which also want to acquire more information but may deviate arbitrarily from the protocol – by employing standard (classical) techniques for boosting the security of Honest-but-Curious protocols to the Malicious adversarial setting (e.g. using the GMW compiler as in [61]). The field has since developed extensively in various settings (see [29] and references therein).

The quantum analogue (Multi-Party Quantum Computation, or MPQC) involves the computation of a transformation on classical or quantum inputs using a quantum computer and has attracted a lot of attention as a potential key application for quantum networks [109] through its ability to preserve privacy and integrity of the highly valuable computations quantum computers would enable. The case of multiple parties was addressed in [14, 30] where an honest majority of players was required, while Two-Party Quantum Computing (or 2PQC) is the focus of [44] for quantum Honest-but-Curious Adversaries and [45] for quantum Malicious Adversaries.

In the early days of quantum computation, researchers believed that quantum capabilities could lead to cryptographic breakthroughs by building various classical multi-party cryptographic primitives in an unconditionally secure way (i.e. without relying on any assumptions beyond the rules of physics and mathematics, thereby offering protection regardless of the computational power of the adversary)[1]. In particular, based on the idea of conjugate coding of [132][2] where two different quantum bases are used to encode information so that no one but the creator can perfectly recreate the original message, first [15] and later [48] proposed schemes for unconditionally-secure Key Distribution[3] requiring only an authenticated classical channel and an insecure quantum channel. The BB84 scheme [15] is as of now implemented on a variety of commercially-available systems. These results sparked hope that many more cryptographic primitives could be uplifted to unconditional security by using quantum systems.

In fact, this has proven to be true for some primitives. In the case of coin flipping, where two parties wish to sample unbiased randomness together, it is possible to construct unconditionally-secure strong coin-flipping protocols with biases as low as $\epsilon = 1/4$ (i.e. no party can skew the result either way with probability higher than ϵ) as demonstrated by various protocols starting with the one presented in [6], and it has been proven in [2][4] that there exist unconditionally-secure protocols for weak coin-flipping (contrary to the strong variant above, each party here has a preferred outcome towards which it wants to bias the coin) for any arbitrarily-low bias. Both of these are impossible in the classical setting, yet such protocols have already been implemented on real quantum hardware [110].

A flurry of unconditionally-secure protocols for the more advanced bit-commitment functionality[5] have been proposed around the same time, for example [19]. Unfortunately, a flaw was discovered which allows an adversary to completely break this protocol. Patches and attacks were exchanged until a series of no-go theorems were finally proven, first showing that unconditionally-secure bit commitment is impossible [95, 100], and later that the same held for oblivious transfer[6] as well [93]. In the end, [121] showed that any non-trivial functionality[7] for two-party computations necessarily leaks some information to the adversarial party. Since then it is established that any protocol for non-trivial functionalities is either only computationally secure (where the protocol is secure so long as the adversary is incapable of solving a given hard problem, meaning that, given enough time and computational power, an adversary

[1] This is also called statistical security.

[2] Submitted to IEEE in 1970 but rejected.

[3] This functionality allows two player to sample a random common key in a way that prevents external eavesdroppers from learning information about this shared secret.

[4] This is a simplification of a previous proof from [103], which has not been published.

[5] It allows one player to send an encrypted message to another such that the sender, if asked to show the clear-text, cannot reveal anything but the original message.

[6] One party proposes two bits to another, who can choose and receive only one without gaining any knowledge about the other. The sender does not learn which one has been chosen.

[7] These are functionalities for which there exists no trivial protocol in the Honest-but-Curious setting, the trivial protocol being defined as follows: one party generates the output and sends it to the other party.

will be able to break the scheme) or requires the pre-existence of certain secure simple cryptographic primitives.

This is a trajectory that is often followed by newly developing fields of research, where initial intuitions built upon ground-breaking early results (Quantum Key Distribution in this case) turn out to be wrong for subtle reasons that appear only through increasing levels of formalisation. In that period, methods are often used without a proper framework for analysing the full breadth of consequences that ensue from their application. That is not to say that such works are merit-less as they are precisely the reason why further axiomatisations and formal models emerge, yielding precise definitions that are then much more useful for building future techniques.

This formalisation endeavour has already been pursued extensively in the case of classical cryptography, where initial game-based proofs have been shown to offer insufficient levels of security in the context of networked machines executing a large number of processes and protocols in parallel. For example, a message could be sent by a party in a given instance of a protocol and later reused by the receiving party in another instance of the same protocol. This is not captured by some of the basic security definitions and concrete protocol attacks have been devised by using this principle. This lead to the development of heavier but more general security frameworks that seek to incorporate these previously overlooked behaviours and loopholes due to the potentially arbitrary composition of protocols. Compared to the time-line of classical cryptography, these formalisation efforts can be seen as rather recent (for instance the framework of Universally Composable Security, or UC, was proposed only in 2001 by [25]) and more results are being published to this day to augment the capabilities of these models.[8]

Concurrently, various efforts have been seeking to translate these formalisations to the quantum case. The work of [126] defines the Quantum Universally Composable (Q-UC) framework of security, the quantum equivalent of the above-mentioned UC, and proves two important and intuitive results. The first one states that any protocol which is statistically-secure in UC is also statistically-secure in Q-UC, i.e. quantum power does not help an adversary in the statistically-secure setting in UC. This is intuitive in the sense that the computational power of unbounded classical and quantum adversaries is the same (the set of computable functions does not change). Then [126] also proves that this holds as well for computational security so long as the hardness assumption upon which the protocol is based remains valid for quantum adversaries. This again seems intuitive. However care must be taken since some properties of classical adversaries do not translate directly to the quantum case, for example copying the internal state of the adversary is no longer allowed due to the so-called No-Cloning Theorem of quantum mechanics. Therefore, while they may be intuitive, these results are far from trivial.

Building upon this line of work, [49, 42] answer the question of what more can be done by using quantum resources compared to classical protocols in Q-UC, as the result above merely performed the translation in one way. Together they categorise all classical functionalities hierarchically such that those that are higher can be used as basis for a secure protocol that implement lower functionalities. Compared to the classical case, this hierarchy is much simpler both in the computational and statistical setting, meaning that quantum operations increase the power of some functionalities in the sense that they can now construct more complex functionalities than they would be able to using only classical communications and processing. This is an extension of an early result from [31] which shows that

[8] Most notably in order to circumvent various no-go theorems that preclude the existence of some functionalities or invalidate the security of intuitive protocols without presenting attacks, both of which may be analysed as artefacts of the model rather than issues regarding the objects themselves.

the bit commitment primitive can be used with a quantum channel to implement the classically much stronger oblivious transfer primitive,[9] which is impossible using only classical communication channels.

These results essentially mean that the proofs of security which are valid in UC also produce a quantum-secure protocol and furthermore that quantum capabilities boost some functionalities to levels that are unattainable classically. Therefore the problematic behaviours that could arise from replacing a classical adversary with a quantum one in fact do not happen in the context of composably-secure frameworks and proofs. This does not however translate directly to all constructions and all security frameworks, as other problems emerge in stand-alone security framework which allows more leeway in the methods used in security proofs. One technique that does not work as in the classical case is the ability to rewind the adversary: if the adversary is classical, it can be seen as a machine whose state can be saved at some point of the protocol and then made to perform the protocol on multiple execution branches by running it multiple times starting from this state snapshot. Transposing this to the quantum case (with a quantum state) is not possible due to the above-mentioned no-cloning result. One may wonder if this is an issue concerning protocols (i.e. does the inability to use this proof technique in the quantum case result in a concrete quantum attack on protocols that use it in their classical security proof?) or whether the quantum security framework simply needs to be expanded (i.e. is there an equivalent quantum technique that can be used for the same purpose and substitute rewinding in quantum proofs?). The work of [7] suggests that attacks may be possible since there exists an oracle relative to which some classically-secure protocols that use rewinding are broken.[10] On the other hand, the results from [127, 129, 128] show that imposing additional constraints on the protocol allows for recovering some previous classical security results. More recently, the Fiat-Shamir construction [51], a classical technique used to transform zero-knowledge proofs of knowledge into signatures in the random oracle model, proven secure against classical adversaries in [113], has been lifted to quantum security in [92, 38].

Many other purely quantum protocols cannot rely on classical proof techniques for their security and must reinvent both the definitions and the tools used to prove them. To cite but a few, *Quantum Noisy-Storage assumptions* can be used to model a quantum adversary with limited or faulty quantum memory (which is not an absurd assumption by today's standard of quantum hardware), in which case unconditional security is possible for all primitives [131][11]; *Quantum Physical Uncloneable Functions* forbid the existence of two copies of the same quantum object [125, 108], thus providing strong authentication guarantees [39]; *Relativistic Cryptography* combines special relativity and quantum mechanics to provide unconditionally-secure bit commitment (it has been proven that this cannot be composably secure unfortunately so the security of protocols in this framework must be proven from scratch every time); finally *Device-Independent Quantum Cryptography* provides security guarantees even in the case where the devices used to perform protocols are not trusted (supplied by the adversary for example).

But in order for these new quantum cryptographic ambitions to materialise, novel definitions and

[9] This primitive is classically universal in the sense that it can be used to implement any other with a polynomial number of calls [79].

[10] The "oracle" simply provides the adversary information in the form of a state that does not help it in the classical case but allows it to break the protocol if it has quantum capabilities.

[11] Their protocols also do not require honest players to have any quantum memory, which is strictly better than the classical case where they need a memory size equal to the square root of the adversary's memory [46] (on top of the fact that classical memory is cheap and so the bound must be taken to be relatively large for security to hold).

proof techniques need to be developed that both precisely capture the full breadth of their possibilities, while at the same time eschewing the caveats and pitfalls of previous intuitive yet flawed arguments of security. In this context, the goal of this thesis is twofold. The purpose of its first part, comprising the first two results, is to forge ahead in this pursuit of precision by filling in some of the gaps previously left open. The latter chapters and results then focus on effectively bringing to life secure quantum protocols on already existing hardware, by lowering overheads in their implementations and tailoring them to the specificities of experimental executions, a task which unfortunately still remains all too often a blind spot in protocol design.

The Oxford Dictionary gives the following definition of *folklore*: "the traditions and stories of a country or community". The particular (and peculiar) community of cryptographers has a number of such tales, most often in the form of informal insights, passed down orally from thesis adviser to PhD student. These instruction (or lār in Old English) are most often accurate and useful in the construction of secure protocols. However, when they fail it is often hard to point out where and why, precisely due to the fact that their presentation does not come with formal statements regarding their security capabilities. On the other hand, most proofs of security are formalised in what is referred to as the "ideal/real world paradigm", where the ideal world consists of perfectly secure protocols (with trusted third parties doing the dirty work), the real world is that of protocols that are actually implementable trustlessly, and the proof shows the equivalence of these two types of execution. We start by uplifting a folklore technique called Cut-and-Choose, which is used to guarantee the correctness of a message without divulging it, by giving its formalisation as an ideal functionality. This includes an extension to the quantum case called Quantum Cut-and-Choose. Prior misapplications of this technique have lead to concrete attacks and subsequent patches to multiple protocols, yet no formalisation of its security properties has been given up to now.

Another problem that lacked proper treatment is that of attacks that occur when an adversary gains quantum access to a classical primitive or protocol, in the sense that any previously classical messages are replaced by quantum states. Previous analyses focused on the case of unconditionally secure protocols and proved their insecurity by showing that any such protocol necessarily leaks information. The reasoning went that if those perfect protocols were insecure, surely those relying on computational assumptions were as well. We disprove this line of thought by building an ideal/real framework of computational security against so-called superposition attacks and constructing secure protocols in this setting. This is done by first dissecting a new superposition attack on the well-known classical Yao protocol for secure two-party computations. This attack allows an adversary to obtain the XOR of two legitimate outputs (while a secure protocol would only return one). Analysing the finer points where the protocols fails yields the following interesting result: the output does not even need to be returned by the honest player to the adversary for the attack to work and the attack vector turns out to be a message that in the classical setting the adversary is supposed to already know.

After bringing these previously informal analyses to the more rigorous world of the ideal/real world paradigm of security, the final two results aim to bring cryptographic protocols to the actual real world. This goal was inspired by talks with various experimental groups, amongst which the ones lead by the Quantum Internet Alliance which seeks to develop the quantum equivalent of the Open Systems Interconnection model of classical networks, therefore allowing a full-stack integration between quantum protocols and hardware. Achieving this requires first simplifying the steps of the protocols that need to

be later implemented and reducing the overhead of the operations performed by most of the participating devices. We devise for this purpose a new protocol for the general Multi-Party Quantum Computation functionality which improves on almost all known efficiency metrics and introduce novel functionalities along the way that may be of independent interest for designing future protocols.

Finally, going even further towards an actual protocol implementation, we focus in the last section on creating protocols for devices that will be available in the near future but may suffer from high levels of noise, for the specific task of Verifiable Delegated Quantum Computations (or VDQC, where a client wants to delegated a quantum computation to a powerful server but does not trust it regarding the data privacy or output integrity). While many other protocols have been devised for this functionality, they are unusable on inherently noisy devices since the noise will trigger the same defence mechanisms as adversarial deviations by the Server, therefore signalling to the client that it must abort. The description of our protocol is voluntarily low-level, taking into consideration communication steps which are absent from previous papers, specifically for the purpose of helping with its future implementation since ambiguity in a protocol's specification can lead to over-optimisation later on, some of which may break its security guarantees.[12] In the process of proving the security of our protocol, thereby closing the thematic circle started with our first results, we formalise for the first time the proof of security of a commonly-used technique: if the output of a protocol is deterministic, it is possible, without breaking the security of the protocol, to repeat computations and later perform classical errors-correction by taking the majority output. In classical cryptography, this fact has been known for a long time and, legitimately, used in various protocols to boost their acceptance probability. However, its application to quantum protocols is not straight-forward since attacks that may be entangled across rounds of execution need to be taken into account.[13] This technique is mentioned in a few papers that deal with VDQC but are rarely if ever formally justified. Our proof of security demonstrates that, rather than being easily brushed off as a trivial extension of the classical case, this method requires precise handling in order for it to be analysed in detail.

These result are now presented in more detail in the section below, along with an outline of the thesis.

Thesis Results and Outline

This thesis consists of two introductory chapters and four research chapters.

Chapter 2 introduces the definitions and results that will form the mathematical basis of the rest of the thesis. We first describe there elements of probability theory that help us analyse the success rates of our protocols. Then we present notions of Quantum Information Theory and Measurement-Based Quantum Computation (MBQC), which has been first described in [118] and will be the main quantum computational framework used in this thesis.

Chapter 3 introduces cryptographic notions that are used in the thesis, starting with a description of basic functionalities, among which Bit-Commitment. We then explain how parties participating in protocols are modelled in frameworks of quantum cryptographic security and present the Quantum Stand-Alone Model of security [66] and later the Abstract Cryptography framework [99], detailing

[12] For instance, optimising some operations in encryption schemes may lead to timing side-channel attacks.

[13] These attacks are more powerful than simple classical correlations between protocol rounds that may happen in classical cryptography.

their various properties along the way. Various Ideal Functionalities which are compatible with both frameworks are then formalised. Then we describe the security properties of a specific class of protocols called Delegated Quantum Computation (DQC) Protocols. Finally, we present other cryptographic primitives used in later chapters of the thesis: two important DQC Protocols derived from MBQC, namely Universal Blind Quantum Computation (UBQC) [21] and Verifiable Blind Quantum Computation (VBQC) [78], the classical Yao Protocol [135] and the Universal Thresholdiser [16].

Chapter 4: Quantum Cut-and-Choose. We start by looking at ways to define precisely the security guarantees of a classical technique that has been informally used in numerous protocols, namely Cut-and-Choose (CC). It is a standard method used to boost the security of a protocol secure against Honest-but-Curious Adversaries to being secure against fully Malicious ones, by essentially enforcing the Honest-but-Curious behaviour. Abstractly, the party sending a message to be secured via CC is made to produce multiple versions of this message with independent randomness (the cutting part) and later on the receiving party can choose a certain number of messages (usually all but one or half of the prepared messages) for which the randomness used in their creation will be revealed (the choosing part). The receiving party can then check that these opened messages have been prepared according to the specifications of the protocol. If these checks pass, the receiver is assured with high probability that the unopened messages have been similarly honestly generated and can therefore be used without risk in the rest of the protocol.

We extend this technique to the quantum case, yielding the Quantum Cut-and-Choose (Q-CC), and at the same time present the first known formalisation of it in the ideal/real world paradigm of security (Stand-Alone Model of [66]). This forces us to precisely define the security notion that the Q-CC protocol achieves in the form of an Ideal Functionality representing the best case scenario of a cryptographic protocol where the parties send their inputs privately to a trusted third party which then returns the outputs. Our Send Blind Correct State Functionality receives from the Sender a state, classical message and proof of correctness, checks that they are correct and then transmits only the state and message to the Receiver. We show that the Q-CC protocol is secure in the sense that it emulates this functionality (meaning that an Adversary can learn nothing beyond what it gets in the ideal scenario), but only if the state, message and proof satisfy a set of requirements. The application and analysis of the Cut-and-Choose technique in protocols secure against quantum Adversaries is not a straightforward transposition of the classical case, among other reasons due to the difficulty to use "rewinding" in the quantum realm. The security proofs we present use novel adaptations of two quantum rewinding techniques, namely Watrous' oblivious quantum rewinding [130] and Unruh's special quantum rewinding [127, 129].

When restricted to purely classical messages, we describe another functionality, the Send Blind Correct Fraction of Messages, whose aim is to multiple message with the guarantee that at least a given fraction of those have been prepared correctly. An outer protocol relying on this functionality as a subroutine has then to perform some form of error-correction to mitigate the effect of this malicious fraction. The corresponding Fraction Classical Cut-and-Choose Protocol is proven secure in the fully composable Abstract Cryptography framework of [99] under another set of assumption, thus providing additional flexibility in how to implement CC. We further optimise the ratio of tested messages to tailor it precisely to the desired maximal ratio of incorrect forwarded messages. This yields interesting

results when applied to the most common use-case for this technique: when a majority vote is used as error-correction, therefore requiring that at least half of the message be correct, the optimal fraction of tested message turns out to be 3/5, when all protocols up to now use the (more intuitive) value 1/2. All such protocols benefit from switching to this new value at no additional cost.

Although the Quantum Cut-and-Choose functionality is useful on its own, we furthermore describe a Compiler based on it which takes as input a protocol secure against a weak *Semi-Malicious* Adversary, which is only allowed to cheat in a subset of messages of the protocol, and outputs a protocol secure against fully Malicious Adversaries. The requirements for a protocol to be Cut-and-Choosable (ie. able to undergo the transformation described in our Compiler) are also fully characterised, which opens the possibility of applying the Q-CC technique to any quantum protocol that fulfils these criteria. The Q-CC technique and Compiler are then applied to a protocol for securely performing a Two-Party Quantum Computation with classical inputs and quantum outputs. The concept of secure delegated quantum computing [21] is used as basis, and in particular the protocol for quantum garbled circuit computation of [77] that has been only proven secure against weak quantum Honest-but-Curious Adversaries. The compiled protocol achieves the same functionality as in previous works on secure Two-Party Quantum Computing such as [45], however using the Quantum Cut-and-Choose technique on the protocol from [77] leads to the following key improvements: (i) only two rounds of quantum communication are necessary (one of which is offline); (ii) only one party needs to have involved quantum technological abilities; (iii) only minimal extra cryptographic primitives are required, namely one Oblivious Transfer for each input bit and quantum-safe commitments; (iv) these primitives can be efficiently instantiated using [112, 128] respectively to obtain one-sided statistical security, thus retaining this same security property of [77].

Chapter 5: Computational Resistance to Superposition Attacks. Recent advances in quantum technologies threaten the security of many widely-deployed cryptographic primitives if we assume that the Adversary has classical access to the primitive but can locally perform quantum computations. This scenario has led to the emergence of *post-quantum cryptography*. But the situation is even worse in the *fully quantum* scenario, if we assume the Adversary further has quantum access to the primitive and can query the oracle with quantum states in superposition. Such access can arise in the case where the Adversary has direct access to the primitive that is being implemented (eg. symmetric encryption, hash functions), or if a protocol is used as a sub-routine where the Adversary plays all roles (as in the Fiat-Shamir transform based on Sigma Protocols) and can therefore implement them all quantumly. In the future, various primitives might natively be implemented on quantum machines and networks, either to benefit from speed-ups or because the rest of the protocol is inherently quantum. In this case, more information could be leaked, leading to new non-trivial attacks, as presented in a series of work initiated in [33, 17, 70]. A possible countermeasure against such *superposition attacks* is to forbid any kind of quantum access to the oracle through measurements. However, the security would then rely on the physical implementation of the measurement tool, which itself could be potentially exploited by a quantum Adversary. Thus, providing security guarantees in the fully quantum model is crucial. We focus here on the multi-party (interactive) setting.

It is of folkloric belief that the security of classical cryptographic protocols is automatically broken if the Adversary is allowed to perform superposition queries and the honest players forced to perform actions coherently on quantum states. Yet another widely held intuition is that disturbing the exchanged

messages via any measurement is enough to protect protocols from these attacks.

However, the reality is much more complex. Security models dealing with superposition attacks only consider unconditional security. Conversely, security models considering computational security assume that all supposedly classical messages are *measured in the computational basis*, which forbids by construction the analysis of superposition attacks. To fill in the gap between those models, the seminal work of [17] started to study the quantum computational security for classical primitives, but only in the single-party setting. To the best of our knowledge, an equivalent model in the multi-party setting is still missing.

In this chapter, we propose the first computational security model considering superposition attacks for multi-party protocols. We show that our new security model is satisfiable by proving the security of the well-known One-Time-Pad protocol and give an attack on a variant of the equally reputable Yao Protocol for Secure Two-Party Computations. The post-mortem of this attack reveals the precise points of failure, yielding highly counter-intuitive results: Adding extra classical communication, which is harmless for classical security, can make the protocol become subject to superposition attacks. We use this newly imparted knowledge to construct the first concrete protocol for Secure Two-Party Computation that is resistant to superposition attacks. Our results show that there is no straightforward answer to provide for either the vulnerabilities of classical protocols to superposition attacks or the adapted countermeasures.

Chapter 6: Quantum Round-Optimal Protocol for Delegated Multi-Party Quantum Computations versus Dishonest Majority. Contributing to the latest challenges in the field of MPQC, alongside recent results from [40, 90, 5], we present a new efficient protocol achieving security even in the case of a single honest Client in the fully composable Abstract Cryptography Framework [99]. The security of our protocol is reduced, in an information-theoretically secure way, to that of a classical composable SMPC used to coordinate the various parties. Our scheme thus provides a statistically secure quantum upgrade of such classical schemes.

In addition, (i) the Clients can delegate their computation to a powerful fully fault-tolerant Server and only need to perform single qubit operations to unlock the full potential of multi-party quantum computation; (ii) the amount of quantum communication with the server is reduced to sending quantum states at the beginning of the computation and receiving the output states at the end, which is optimal and removes the need for interactive quantum communication; and (iii) it has a low constant multiplicative qubit overhead compared to the single-Client delegated protocol it is built upon [78].

We bootstrap the MPQC construction using a new composable resource called Double Blind Quantum Computation. It replaces the state preparation step of the single-client delegated protocol by a multi-party state preparation in a way that allows us to then drive all required coordination tasks between the parties by a Classical SMPC scheme. Crucially, we only use this resource to perform *constant-depth blind but not verifiable* quantum computations. Beyond the obvious efficiency gains, this goes on to show that, contrary to previous beliefs, verification of the full protocol can be achieved without requiring verifiability of all components.

Similar to matryoshka dolls, the new Double Blind Quantum Computation Protocol implementing this resource is based off of the Universal Blind Quantum Computation Protocol of [21], where the qubit communication between Client and Server has been replaced by a collaborative Rotated State

Preparation Protocol which ensures that no malicious coalition has any information about the generated state so long as the honest Client's secret is not revealed. We further describe another such State Preparation Protocol used for preparing collaboratively state from the BB84 set. Beyond their usefulness in constructing the MPQC Protocol, these three sub-protocols are of independent interest and may find uses in constructing other multi-party functionalities.

Chapter 7: Qubit and Operation Optimal Verifiable Quantum Computations. Going even further towards concrete implementations of complex quantum protocols, we finally focus on tailoring a specific cryptographic protocol to experimental constraints. Recent achievements have shown that experimental quantum computers can outperform their classical counterparts. This motivated an ever growing interest from companies starting to feel the limitations of classical computing. Yet, in light of the ongoing stream of privacy scandals, the current commercial developments pointing toward the future availability of quantum computing through remotely accessible servers poses peculiar challenges: Clients, with limited quantum capabilities, will want their data and their algorithms to remain hidden from the servers, while still being able to verify that their computations are being performed correctly.

Numerous theoretical protocols for delegating quantum computations in a secure (blind and verifiable) way have been proposed over the years in an attempt to address the question. However, few have been implemented on existing hardware as not only do all currently available techniques suffer from high qubit overheads but also, and more importantly, from over-sensitivity: When running on noisy devices, plain imperfections trigger the same detection mechanisms as malicious attacks, resulting in perpetually aborted computations. Therefore, although malicious quantum computers are rendered harmless by blind and verifiable protocols, their inherent noise severely limits their usability.

We address this problem by introducing an efficient, robust, blind, verifiable Measurement-Based Quantum Computation scheme allowing a client with limited quantum capabilities to perform delegated deterministic computations with classical inputs and outputs. We assume for this purpose the following setup. The client, which has only the ability to rotate single qubits around the Z-axis with a restricted set of angles and perform X-basis measurements, chooses an MBQC computation with deterministic classical output to be delegated to a quantum server of known topology. The server's device running this computation is inherently noisy in the sense that it will fail to return the correct measurement outcomes with some probability which is upper-bounded by a constant. Our protocol transforms the client's computation into a blind computation that can be run on the same devices with exponential success probability and which offers exponential verification against a malicious server. This is done by leveraging the deterministic nature of the computation and repeating the runs of computation with different hiding parameters before taking a majority outcome, which results in a high acceptance probability on these devices. Test runs are introduced at random among the computation runs to detect a deviating server.

We prove the security of the protocol again in the Abstract Cryptography Framework as the emulation of the Verifiable Delegated Quantum Computation Resource, using the reduction to local criteria from [41]. More precisely, we show that (i) a fully malicious server cannot learn any information about the computation and inputs and can cheat at most with an exponentially small success probability; (ii) it is robust in the sense that, in presence of sufficiently low non-malicious noise, the protocol will succeed with a probability exponentially close to 1; (iii) the induced overhead is limited to a polynomial number of repetitions of the initial computation interleaved with test runs of similar complexity,

i.e. requiring the same physical resources per run in terms of memory and gates, in particular the required size of the quantum device remains exactly the same as in the initial MBQC computation; (iv) the amount of tolerable noise, measured by the probability of failing a test run, can be as high as 25% for some computations and will be generally bounded by 12.5% when the computation is implemented using a planar graph resource state.

The key ingredients used in this work are both the realization that security can be provided without using universal computation graphs and that, in our setting, we do need to use fault-tolerance for the purpose of amplifying the acceptance nor security level exponentially close to 1. While our protocol is most useful in the regime where fault-tolerance can be used to lower the noise level below our threshold in point (iv) above, this work is presented in an iterative way so that it may be readily implemented as a proof-of-concept on limited devices of a few qubits such as those currently being developed by the Quantum Internet Alliance, which satisfy our noise constraints as well. The code is identical for all execution sizes and can be directly reused when larger devices are created. This allows us to demonstrate the feasibility of our protocol at a time when other realisations of the same functionality are unthinkable due to their prohibitive memory overheads.

Publications

- The results forming the basis of this thesis may be found in the following works:

[106] "Dispelling Myths on Superposition Attacks: Formal Security Model and Attack Analyses",
L. Music, C. Chevalier, E. Kashefi,
Proceedings of Provable and Practical Security (ProvSec 2020).

[73] "Delegating Multi-Party Quantum Computations vs. Dishonest Majority in Two Quantum Rounds",
T. Kapourniotis, E. Kashefi, L. Music, H. Ollivier,
Arxiv Pre-Print

[87] "Verifying Quantum Computations on Noisy Devices with Minimal Overhead",
E. Kashefi, D. Leichtle, L. Music, H. Ollivier,
Arxiv Pre-Print

[75] "The Quantum Cut-and-Choose Technique and Quantum Two-Party Computation",
E. Kashefi, L. Music, P. Wallden,
Arxiv Pre-Print

- The following works are in preparation:

"Quantum Protocol Compiler: Boosting Security from Semi-Malicious to Malicious Adversaries",
E. Kashefi, L. Music, D.Unruh, P. Wallden

"Fully-Composable Quantum Exponential Security Of Classical Cut-and-Choose",
L. Music, C. Chevalier, E. Kashefi

"Verification of BQP and Sampling on Noisy Devices",
D. Leichtle, L. Music, E. Kashefi, H. Ollivier

"Approaching Qubit Optimality in MPQC",
T. Kapourniotis, E. Kashefi, L. Music, H. Ollivier

Chapter 2

Preliminaries in Probability Theory and Quantum Information

Elements of probability theory and quantum information are recalled here, along the quantum computational framework that forms the basis of the protocols presented later. Basic general notations which will be used in the rest of the thesis are first presented:

- for any positive integer $n \in \mathbb{N}^*$, $[n] := \{1, \ldots, n\}$;
- for any integers n and k, $\binom{n}{k}$ designates the number of k-combination of a set of size n;
- $\#X$ is the size of X (e.g. string length, set size, number of qubits in a quantum register);
- $n!$ is the factorial of $n \in \mathbb{N}$, which satisfies Sterling's approximation $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$;
- for any set S, $\wp(S)$ is the set of subsets of S;
- $\oplus$ and $\odot$ are respectively the bit-wise XOR and AND operations on bit-strings of the same length;
- $\|$ represents string concatenation;
- $\mathrm{w_H}(\mathbf{s})$ is the Hamming weight of string $\mathbf{s}$;
- for any two binary strings $\mathbf{a}$ and $\mathbf{b}$ of length n, $\mathbf{a} \cdot \mathbf{b} = \sum_{i \in [n]} a_i b_i \pmod 2$ designates the scalar product of $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{Z}_2^n$.
- $\{0,1\}^*$ is the set of all bit-strings;
- $\wedge$ and $\vee$ represent respectively the logical AND and OR operations;
- $x \in_R X$ means that the value x is sampled from set X uniformly at random;
- for distribution D, $x \leftarrow D$ denotes that x was sampled according to distribution D (we abuse notation and use $\leftarrow$ more generally to indicate that a variable is initialised with the result of an operation);
- if $X \sim D$ then the random variable X follows distribution D;
- for any well-defined event A, we denote $\Pr[A]$ its probability;
- for any random variable X, $E(X)$ is its expected value;
- for a set V and $O \subseteq V$, $O^c := V \setminus O$ (this should depend on the set V in all generality but will be

clear from context);
- for a graph G, $N_G(v)$ are the neighbours of vertex v in graph G.

2.1 Useful Inequalities from Probability Theory

We assume familiarity with notions of probability theory, see [50] for more material. The following definitions and lemmata regarding two probability distributions are useful to derive bounds on the security of protocols.

2.1.1 Binomial Distribution

A random variable X follows the *binomial distribution* if for example it counts the number of successful results of a trial with boolean results which is independently repeated m times such that each try is successful with constant probability p, or equivalently, the number of marked items drawn after drawing m items from a set with a fraction p of marked items *with replacement*.

Definition 2.1 (Binomial Distribution)**.** *Let $m \in \mathbb{N}^*$ and $p \in (0,1)$. A random variable X is said to follow the* binomial distribution*, denoted as $X \sim \mathrm{Bin}(m,p)$, if for $0 \leq k \leq m$ its probability mass function is described by:*

$$\Pr[X = k] = \binom{m}{k} p^k (1-p)^{m-k} \tag{2.1}$$

We now give the result of the application of *Hoeffding's Inequality* to the binomial distribution defined above.

Lemma 2.1 (Hoeffding's Inequality for the Binomial Distribution)**.** *Let $X \sim \mathrm{Bin}(m,p)$ be a random variable. For any $0 \leq k \leq \mathbb{E}[X] = mp$ it then holds that:*

$$\Pr[X \leq k] \leq \exp\left(-2\frac{(mp-k)^2}{m}\right) \tag{2.2}$$

Similarly, for any $k \geq mp$ it holds that:

$$\Pr[X \geq k] \leq \exp\left(-2\frac{(mp-k)^2}{m}\right) \tag{2.3}$$

2.1.2 Hypergeometric Distribution

A random variable X following the *hypergeometric distribution* describes the number of drawn marked items when drawing m items from a set of size n containing d marked items *without replacement*.

Definition 2.2 (Hypergeometric Distribution)**.** *Let $n, d, m \in \mathbb{N}$ with $0 \leq m, d \leq n$. A random variable X is said to follow the* hypergeometric distribution*, denoted as $X \sim \mathrm{Hyp}(n,d,m)$, if for $0 \leq k \leq m$ its probability mass function is described by:*

$$\Pr[X = k] = \frac{\binom{d}{k}\binom{n-d}{m-k}}{\binom{n}{m}} \tag{2.4}$$

We now give a few *tail bounds* and accompanying corollaries associated with the hypergeometric distribution defined above.

Lemma 2.2 (Tail Bound for the Hypergeometric Distribution)**.** *Let $X \sim \mathrm{Hyp}(n, d, m)$ be a random variable and $0 < \epsilon < d/n$. It then holds that:*

$$\Pr\left[X \leq \left(\frac{d}{n} - \epsilon\right) m\right] \leq \exp\left(-2\epsilon^2 m\right) \tag{2.5}$$

Corollary 2.1. *Let $X \sim \mathrm{Hyp}(n, d, m)$ be a random variable and $0 < \lambda < \mathbb{E}[X] = md/n$. It then holds that:*

$$\Pr\left[X \leq \lambda\right] \leq \exp\left(-2m\left(\frac{d}{n} - \frac{\lambda}{m}\right)^2\right) \tag{2.6}$$

Lemma 2.3 (Serfling's Bound for the Hypergeometric Distribution [62, 123])**.** *Let $X \sim \mathrm{Hyp}(n, d, m)$ be a random variable and $\lambda > 0$. It then holds that:*

$$\Pr\left[\sqrt{m}\left(\frac{X}{m} - \frac{n}{d}\right) \geq \lambda\right] \leq \exp\left(-\frac{2\lambda^2}{1 - \frac{m-1}{n}}\right) \tag{2.7}$$

Corollary 2.2. *Let $X \sim \mathrm{Hyp}(n, d, m)$ be a random variable and $\lambda > md/n$. It then holds that:*

$$\Pr\left[X \geq \lambda\right] \leq \exp\left(-2m\left(\frac{\lambda}{m} - \frac{d}{n}\right)^2\right) \tag{2.8}$$

Note the symmetry of Corollary 2.1 and Corollary 2.2.

2.2 Quantum Information Theory

We give here a brief overview of the definitions and results from Quantum Information Theory that are useful in this thesis. We refer the reader to [107] for more details.

This theory describes the behaviour of information when it is encoded in the properties of particles and systems governed by the laws of quantum mechanics. The most basic system described by quantum mechanics is one with two possible "basic" states, for example the excited and non-excited states of the hydrogen's electron, or the horizontal and vertical polarisations of a photon. While a system is either in one state or the other in the classical case (although this can be chosen according to a probabilistic distribution), it can be in both state at the same time according to the laws of quantum mechanics. The object is then said to be in superposition.

However, making the observation on a quantum system will break this superposition and make the system collapse to one state or the other. Before any measurement however, the evolution of the system behaves exactly as if both states were present at the same time, which yields counter-intuitive results such as being able to perform the two-slit experiment with particles instead of waves. In this case, each of the particles traverse both of the slits at the same time and are only measured when they arrive on the screen on the other side. This phenomenon is called quantum interference and can be leveraged to boost the speed of computations for instance when using quantum properties for information processing.

On the other hand, if we try to measure through which slit the particle has passed, we find that there is only ever a single particle (the detectors either detect it on the left or right slit but never at both) but then the interference pattern disappears from the screen since the first measurement has destroyed the superposition and forced the particles to behaves as classical objects. It is another fundamental property of quantum mechanics that there are measurements which are incompatible with one another: performing one before the other will change the result. This gives rise to the *uncertainty principle* stating that not all properties of an object that follows the laws of quantum mechanics can be measured at the same time.

It is therefore not possible to replicate exactly an arbitrary quantum system by trying to determine its characteristics via measurements. More interestingly still, this is impossible even when purely quantum operations are allowed. This result takes the form of the *no-cloning theorem*, which has far-reaching applications in cryptography for instance. One such use-case is Quantum Key Distribution, where bits of a cryptographic key are encoded in the properties of a quantum system. Intuitively, if the recipient of the key has been able to recover it correctly, then it is guaranteed that no one has a copy of it, regardless of the computational power of the adversary. Even more interestingly, any action by an eavesdropper that leads it to acquire some information about the key is equivalent to performing a measurement on the system, which as describe above will disturb the state on the receiver's end of the channel. This can then even be used to detect any suspicious activity on the channel.

The interactions between multiple quantum systems bring even more intriguing powers to devices that are capable of manipulating such objects, in the form of *quantum entanglement*. It is for example possible to create pairs of particles in such a way that measuring both particles in a similar way produces the same effect. This holds regardless of the distance separating the two particles and can produce correlations in measurement outputs that are impossible to recreate classically (at least not without communicating). Such entangled pairs can be used then to transfer the state of a given particle to another particle in a process that is called *quantum teleportation* (note that this does not create a copy of the first particle since a measurement is required and so the no-cloning principle is preserved).

We will now formalise mathematically the notion of a quantum state before moving on to the operations and measurements applicable to such systems. Finally we will present a few results, some of which have been sketched above, that will be useful in the rest of the thesis.

2.2.1 Quantum States

Pure Quantum States. Let $\mathcal{H}$ be a Hilbert space of dimension N and $\{|\psi\rangle_i\}_{i\in[N]}$ be an orthonormal basis of $\mathcal{H}$.[1] The normalised vectors of this space represent the possible states of a quantum system. We therefore define *pure quantum states* in Hilbert space $\mathcal{H}$ as:

$$|\psi\rangle = \sum_{i=1}^{N} c_i |\psi_i\rangle \tag{2.9}$$

where $c_i \in \mathbb{C}$ are coefficients satisfying the relation $\sum_{i=1}^{N} |c_i|^2 = 1$. If there are more than one non-null values for c_i, then the state is said to be in *superposition*. We write † for the *Hermitian conjugate*

[1] Finite dimensional Hilbert spaces will be sufficient for the purpose of this thesis as we study *discrete variable* systems only.

operation and for all pure states $|\psi\rangle$ we write $\langle\psi| := |\psi\rangle^\dagger$. These Hermitian conjugates are used to define projections on quantum states and associated measurement (more precisely, projective measurements). The values $|c_i|^2$ then correspond to the probability that the output of a measurement on state $|\psi\rangle$ along the basis $\{|\psi\rangle_i\}_{i\in[N]}$ produces the i^{th} output. The post-measurement state has in that case *collapsed* to the i^{th} basis state $|\psi_i\rangle$. This is called the *Born rule* and justifies the normalisation condition seen above.

Two-level Quantum Systems. The special case of a two-dimensional Hilbert space will be used often. The pure state of a *qubit*, which is the quantum equivalent of a classical bit of information, has in general the following form: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\{|0\rangle, |1\rangle\}$ are the orthogonal computational-basis vectors of $\mathbb{C}^2$ and $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$. For n qubits, the joint system is given by $\mathbb{C}^{2^n} = \mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2$ for n subspaces, where $\otimes$ designates the *tensor product* of Hilbert spaces.[2] We call *computational basis* on $\mathbb{C}^{2^n}$ the family of classical bit-string states $\mathcal{B}_C = \{|x\rangle \mid x \in \{0,1\}^n\}$. We will use the term quantum register instead of Hilbert space with the same meaning as a classical memory register in a classical computer (as a way to reference specific qubits or subsystems).

Mixed States. More generally, we can define for any pure state $|\psi\rangle$ the associated density matrix $|\psi\rangle\langle\psi|$. Then, given a collection of pure quantum states $\{|\psi_j\rangle\}$ (not necessarily orthogonal) and probabilities $p_j \geq 0$ with $\sum_j p_j = 1$, the associated *mixed quantum state* ρ is the probabilistic mixture given by:

$$\rho = \sum_j p_j |\psi_j\rangle\langle\psi_j| \tag{2.10}$$

Such states arise for instance when an observer does not have in its possession all the sub-systems of the Hilbert space. Each value p_j represent the probability that the state is in fact in the pure state $|\psi_j\rangle$. Let Tr be the *linear trace operation* (sum of diagonal matrix elements), then $\text{Tr}(\rho) = 1$ for any density matrix. These matrices are also positive semi-definite. We denote $D(\mathcal{X})$ the set of all possible quantum states in quantum register $\mathcal{X}$.

Shared Quantum Systems. Given a multi-partite state in Hilbert space $\mathcal{H} = \bigotimes_{k=1}^n \mathcal{H}_k$ (consisting for example of multiple qubits held by a given number of parties), the state held by each party k can be written using the partial trace operator. In the two-party scenario, let ρ_{AB} be a quantum state shared between parties A and B, called *joint state of A and B*. Let Tr_B be the *partial trace* over B's system, defined for any a_i and b_i by:

$$\text{Tr}_B\left(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|\right) = |a_1\rangle\langle a_2| \,\text{Tr}\left(|b_1\rangle\langle b_2|\right) \tag{2.11}$$

Then the state as seen by party A is represented using the *reduced density operator* $\rho_A := \text{Tr}_B(\rho_{AB})$. Using the partial trace to describe parts of a larger system stems from the fact that it is the unique operation that gives correct results for measurement on the partial system. It can easily be extended to

[2] If $\mathcal{H}_1$ and $\mathcal{H}_2$ are two Hilbert spaces of dimensions n_1 and n_2 respectively (finite dimensions are sufficient in our case), $\mathcal{H}_1 \otimes \mathcal{H}_2$ is the Hilbert space of dimension $n_1 n_2$ formed by $|\psi_1\rangle \otimes |\psi_2\rangle$ for $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$. The linear operators on $\mathcal{H}_1 \otimes \mathcal{H}_2$ are $U_1 \otimes U_2$, for linear operators U_i on $\mathcal{H}_i$, and act as $U_1 \otimes U_2 |\psi_1\rangle \otimes |\psi_2\rangle = U_1 |\psi_1\rangle \otimes U_2 |\psi_2\rangle$ and extended through linearity. The inner product is given by $\langle psi_2| \otimes \langle\psi_1|\phi_1\rangle \otimes |\phi_2\rangle = \langle\psi_1|\phi_1\rangle \langle\psi_2|\phi_2\rangle$.

the multi-party case. If such a state over M systems can be written as a product state, it is then called *separable* and is of the form:

$$|\psi\rangle = \bigotimes_{k=1}^{M} |\psi_k\rangle \tag{2.12}$$

It is easy to see that in this case the reduced density operator of each party k is equal to ρ_k (since $\mathrm{Tr}_{[M\setminus\{k\}]}(|\psi\rangle) = |\psi_k\rangle \prod_{i\neq k} \mathrm{Tr}(|\psi_i\rangle) = |\psi_k\rangle$). States that cannot be written in this form are said to be *entangled* and the partial trace over entangled pure systems yields mixed states (no one party has full knowledge of its own system).

Basic Quantum States. The following two state are uniform superposition of the qubit computational basis vectors: $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. For n qubits, we call the state represented by the density matrix $\mathsf{I}/2^n$ the perfectly or totally mixed state. This state is an uniform mixture of all possible classical states and a party whose view of a quantum system is represented by this state has no information about the system. The following state will be henceforth called *EPR-pair*, or maximally-entangled Bell state:

$$\left|\psi^+\right\rangle := \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B) \tag{2.13}$$

If both qubits are both measured in the same basis $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ or $\{|+\rangle\langle +|, |-\rangle\langle -|\}$, they will produce the same result. Such correlations are impossible to achieve classically and occur regardless of the positions of both players. However, applying the partial trace over one of the systems produces the maximally mixed state on the other. Note that this state can be created by applying a CNOT gate to the state $|+\rangle|0\rangle$ with the first qubit as control and the second as target (see Section 2.2.2 for the definition of CNOT).

Trace Distance and Fidelity. The trace distance and fidelity allow us to measure how close two quantum states are. The trace norm distance between states ρ_0 and ρ_1 is given by:

$$\Delta_{\mathrm{Tr}}(\rho_0, \rho_1) := \frac{1}{2}\mathrm{Tr}\,|\rho_0 - \rho_1| \tag{2.14}$$

In the equation above, $|A| := \sqrt{A^\dagger A}$ is the positive square root of $A^\dagger A$, i.e. the semi-definitive positive S such that $S^2 = A^\dagger A$. When $\Delta_{\mathrm{Tr}}(\rho_0, \rho_1) \leq \epsilon$, which we will note $\rho_0 \overset{\epsilon}{\approx} \rho_1$, then any process applied to ρ_0 behaves the same as it would on ρ_1 except with probability at most ϵ.

For sub-normalised states (which can for instance represent parts of a state in superposition), we use the following definitions and properties of the sub-normalised fidelity $\tilde{F}$ and the trace distance $\tilde{\Delta}$ from [41]:

$$\begin{aligned} \tilde{\Delta}_{\mathrm{Tr}}(\rho,\sigma) &\leq \sqrt{1 - \tilde{F}^2(\rho,\sigma)} \\ \tilde{F}(\rho,\sigma) &= F(\rho,\sigma) + \sqrt{(1-\mathrm{Tr}\,\rho)(1-\mathrm{Tr}\,\sigma)} \\ F^2(|\phi\rangle, \sigma) &= \langle\phi|\sigma|\phi\rangle \end{aligned} \tag{2.15}$$

2.2.2 Quantum Operations

Now that properties of quantum states have been defined, we can focus on operations applied to such states, starting with measurements on these states.

Unitary Evolution. In the absence of perturbations, the evolution of a quantum state can be described in the form of a *unitary* U, which correspond to any linear operator on a Hilbert space $\mathcal{A}$ containing the state that satisfies the relation $UU^\dagger = \mathsf{I}_\mathcal{A}$, where I is the identity operator. We will write $U(\rho)$ instead of $U\rho U^\dagger$ for applying unitary U to the mixed state ρ.

Projective Measurements. Let M be a Hermitian operator (meaning that $M^\dagger = M$), also called *quantum observable*. Its real eigenvalues correspond to the possible outcomes of the measurement performed by M. The post-measurement state is given by the eigenvector associated with the output eigenvalue. We focus here on a specific type of measurements described by projection operators, called *projective measurement*.

Let $\{P_m\}$ be projectors satisfying the following conditions (where $\mathsf{I}_\mathcal{A}$ denotes the identity operator on the Hilbert space $\mathcal{A}$):

$$\begin{aligned} &\forall m, P_m^2 = P_m \\ &\forall m, P_m \geq 0 \\ &\sum_m P_m = \mathsf{I}_\mathcal{A} \end{aligned} \tag{2.16}$$

It is simple to verify that $P_{m'}P_m = \delta_{m',m}P_m$, where $\delta_{m,m'}$ is Kronecker's delta. The probability p_m of measuring outcome m given state $|\psi\rangle$ is given by (where $\langle\phi|\psi\rangle$ is the inner-product of the Hilbert space, defined states $|u\rangle$ and $|v\rangle$ from any orthonormal basis by $\langle u|v\rangle = \delta_{u,v}$):

$$p_m = \langle\psi|P_m|\psi\rangle \tag{2.17}$$

The corresponding post-measurement state is then:

$$\frac{P_m\,|\psi\rangle}{\sqrt{p_m}}. \tag{2.18}$$

The measurement operators can be generalised, but projective measurements will be sufficient here as any general measurement can be expressed as a unitary on the state and additional qubits initialised each to the state $|0\rangle$ (called ancillary qubits) followed by a projective measurement.

General Quantum Operations. We can now define the transformations on quantum states in between measurement (of which the unitaries mentioned above are a specific case). Let $L(\mathcal{A})$ be the set of linear mappings from Hilbert space $\mathcal{A}$ to itself. A super-operator $\mathcal{E} : L(\mathcal{A}) \to L(\mathcal{B})$ is called *quantum operation* if it is:

- Completely positive (a positive semi-definite operator upon which such a map is applied remains positive semi-definite);

- Trace-non increasing (applying the map does not increase the trace of the state).

If it is trace-preserving then it is called a CPTP-map. An important result for decomposing quantum operation is Kraus' Theorem.

Theorem 2.1 (Kraus Decomposition)**.** *Any quantum operation $\mathcal{E} : L(\mathcal{A}) \to L(\mathcal{B})$ on state $\rho \in L(\mathcal{A})$ can be decomposed into (non-unique) linear operators $\{K_i\}_{i\in[mn]}$ (called* Kraus operators*) with $m = \dim(\mathcal{A})$ and $n = \dim(\mathcal{B})$, with $\sum_i K_i^\dagger K_i \leq \mathsf{I}$, as:*

$$\mathcal{E}(\rho) = \sum_i K_i \rho K_i^\dagger. \tag{2.19}$$

If furthermore $\sum_i K_i^\dagger K_i = \mathsf{I}$, then $\mathcal{E}$ is trace-preserving.

Purification of Quantum States. For any quantum register $\mathcal{Q}$ and any state ρ_Q is it always possible to define, given another sufficiently large quantum system $\mathcal{R}$ (it is sufficient for it to be of the same size as $\mathcal{Q}$), a pure state $|\phi_{RQ}\rangle$ such that looking at the restriction of the system to register $\mathcal{Q}$ (by tracing out subsystem $\mathcal{R}$) gives ρ_Q. This technique is called *purification*, the register $\mathcal{R}$ is called the reference register, and allows to represent any CPTP-map as a unitary on a larger system.

Diamond Distance between Quantum Operations. The diamond distance on CPTP maps measures the maximal distinguishing probability between two quantum operations acting on the same Hilbert space. This is most useful when bounding the distance between the outputs of two different processes for arbitrary inputs. Let $\mathcal{E}$ and $\mathcal{F}$ be two CPTP maps on n qubits, then the diamond distance is given by:

$$\Delta_\diamond(\mathcal{E}, \mathcal{F}) := \max_\rho \Delta_{\mathrm{Tr}}(\mathcal{E} \otimes \mathsf{I}_n(\rho), \mathcal{F} \otimes \mathsf{I}_n(\rho)) \tag{2.20}$$

In the equation above, the maximisation is done over all density matrices of dimension n^2. We will often drop the indices under Δ as it will be clear from the context. We also overload the notation $\overset{\epsilon}{\approx}$, such that $\Delta(\mathcal{E}, \mathcal{F}) \leq \epsilon$ is noted $\mathcal{E} \overset{\epsilon}{\approx} \mathcal{F}$.

2.2.2.1 Common Quantum Operations

We describe here a few common quantum operation, most notably unitaries that will be later combined together to create more complex operations. By analogy with a classical boolean circuit, these composed unitaries can be written as *quantum circuits*. The basic unitaries presented here will then correspond to *gates* in these circuits, applied to states which form the circuit's inputs and wires in the order given by the circuit. Similarly to classical circuits where a bit can be used to control the application of an operation – i.e. the operation is applied if the bit is set to 1 – some quantum wires may act as a control to unitaries on other wires. This way of representing quantum operations is appropriately called the *circuit model*. A CPTP-map on qubits can always be decomposed into unitaries followed by measurements in the computational basis and there is therefore a correspondence between CPTP-maps and quantum circuits if we allow arbitrary gates.

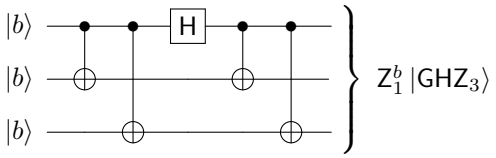
We start by the *Pauli matrices*, which are given by the following equations:

$$\mathsf{X} := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \mathsf{Y} := \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \mathsf{Z} := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.21}$$

Note that Pauli X corresponds to the classical bit-flip operation. The controlled-NOT gate CNOT (with the first qubit being the control) is defined through $\mathsf{CNOT}\,|b\rangle\,|\phi\rangle = |0\rangle\,\mathsf{X}^b\,|\phi\rangle$ for bit b and state $|\phi\rangle$. In the computational basis, this gate XORs the content of the first register to the second register: $\mathsf{CNOT}\,|a\rangle\,|b\rangle = |a\rangle\,|a \oplus b\rangle$ for $a, b \in \{0, 1\}$. The controlled-Z gate CZ is similarly defined using Pauli Z instead of X. The SWAP gate interchanges two quantum registers, i.e. for all states $|\psi\rangle$ and $|\phi\rangle$ we have that $\mathsf{SWAP}\,|\psi\rangle\,|\phi\rangle = |\phi\rangle\,|\psi\rangle$. We call Quantum One-Time Pad (or Q-OTP) the following operation $\mathsf{QOTP}_k(\cdot) = \mathsf{Z}^{k_Z}\mathsf{X}^{k_X}(\cdot)$ for a random Quantum One-Time-Pad key $k = (k_Z, k_X)$ of n bits each, where $\mathsf{X}^{k_X} = \bigotimes_{i=1}^{n} \mathsf{X}_i^{k_X(i)}$ and X_i applies X on qubit i (similarly with Z). It is the equivalent of the classical One-Time-Pad in the sense that any state encrypted using this scheme is *perfectly mixed* to any adversary that does not possess the key – i.e. the adversary can gain no information from the state. This is shown in the following equality for an arbitrary qubit state ρ:

$$\mathrm{Tr}_k(|k\rangle \otimes \mathsf{QOTP}_k(\rho)) = \frac{\mathsf{I}}{2} \tag{2.22}$$

The Hadamard gate is defined by $\mathsf{H}\,|0\rangle = |+\rangle$ and $\mathsf{H}\,|1\rangle = |-\rangle$. We define the logical Hadamard gate H_L by $\mathsf{H}_L\,|0\rangle^{\otimes L} = |+_L\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle^{\otimes L} + |1\rangle^{\otimes L}\Big)$ and $\mathsf{H}_L\,|1\rangle^{\otimes L} = |-_L\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle^{\otimes L} - |1\rangle^{\otimes L}\Big)$. H_L acts as identity on the remaining basis states. The state $|+_L\rangle$ is more commonly referred to as the Greenberger–Horne–Zeilinger $|\mathsf{GHZ}_L\rangle$ state, and $|-_L\rangle = \mathsf{Z}_1\,|\mathsf{GHZ}_L\rangle$, where Z_1 applies a Z operation on the first qubit. The logical Hadamard gate can be constructed by the same circuit as the one generating a GHZ state if we are willing to relax the condition that it acts as identity on the other basis states (it does not affect the results presented in this thesis to use one or the other, since we only apply this gate). The GHZ_L generation procedure starts with an initial state is given by $|0\rangle^{\otimes L}$ and applies a Hadamard gate H on the first qubit and then $L-1$ CNOT gates where the first qubit is the control and qubit $(i+1)$ is controlled by the i^{th} CNOT gate. The result is the state $|\mathsf{GHZ}_L\rangle$. If the initial state is $|1\rangle \otimes |0\rangle^{\otimes(L-1)}$, the resulting state is $\mathsf{Z}_1\,|\mathsf{GHZ}_L\rangle = \frac{|0\rangle^{\otimes L} - |1\rangle^{\otimes L}}{\sqrt{2}}$. If the initial state is $|b\rangle^{\otimes L}$, then by applying $L-1$ CNOT gates where the first qubit is the control and qubit $(i+1)$ is controlled by i^{th} CNOT gate, we recover the state $|b\rangle \otimes |0\rangle^{\otimes(L-1)}$. Putting this together yields the following circuit, which has the desired property (for $L = 3$):



Finally, let $\Theta := \{k\pi/4\}_{k\in\{0,1,\ldots,7\}}$. We define the rotation around the Z-axis associated to $\theta \in \Theta$ as:

$$\mathsf{Z}(\theta) := \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \tag{2.23}$$

Note that $\mathsf{Z}(0) = \mathsf{I}$ and $\mathsf{Z}(\pi) = \mathsf{Z}$, while $Z(\pi/2)$ and $\mathsf{Z}(\pi/4)$ are also called the phase gate P and the T gate respectively. We then also define the following states $|+_\theta\rangle := \mathsf{Z}(\theta)|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$. We call θ-basis measurement the measurement defined by the projectors $\{|+_\theta\rangle\langle+_\theta|, |-_\theta\rangle\langle-_\theta|\}$. Note that an EPR-pair $|\psi^+\rangle$ can be rewritten using states of the form $|+_\theta\rangle$ for any θ in the following way:

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|+_\theta\rangle_A \otimes |+_{-\theta}\rangle_B + |-_\theta\rangle_A \otimes |-_{-\theta}\rangle_B) \tag{2.24}$$

Importantly, if the state on one side is measured in the θ-basis, the state on the other side will collapse in the basis associated with $(-\theta)$.

There exist finite sets of gates – called *universal sets of gates* – such that it is possible to efficiently approximate any efficient CPTP-map with a circuit using only gates from this set. This can be shown by first proving that it is possible to perfectly decompose any unitary into a sequence of single-qubit gates and two-qubit gates, then that the two-qubit gates can be fixed to be CNOT gates only. Then it is possible to show that the group generated by the single-qubit unitaries H, $\mathsf{Z}(\pi/2)$, $\mathsf{Z}(\pi/4)$ is dense in the set of all single-qubit unitaries. Finally, we can use the following theorem from [81] which gives the efficiency argument for any such dense group:

Theorem 2.2 (Solovay-Kitaev Approximation)**.** *Let $SU(2)$ be the set of all single-qubit unitaries with determinant equal to 1.*[3] *Let $\mathcal{G}$ be a finite set of elements of $SU(2)$ such that:*

- *It is possible to produce the inverses of elements of $\mathcal{G}$ by finitely iterating over elements of $\mathcal{G}$.*
- *The group generated by iterating elements of $\mathcal{G}$ is dense in $SU(2)$.*

Consider some $\epsilon > 0$. Then there is a constant c such that for any $U \in SU(2)$, there is a sequence S of gates from $\mathcal{G}$ of length $\mathcal{O}(\log^c(1/\epsilon))$ such that $\Delta(S - U) \leq \epsilon$, where Δ is the trace distance.

2.2.2.2 Simulating Classical Circuits

It is possible to represent any classical operation using a quantum implementation of the reversible classical Toffoli gate computing the function $T(a, b, c) = (a \cdot b) \oplus c$ where $(\oplus, \cdot)$ are defined in $\mathbb{Z}_2$. This operation can be defined as a unitary on three qubits (any reversible classical gate is simply a permutation of the computational basis states) and is universal for classical computations since it can be used to implement the classical NAND gate. Any binary function $f : \{0,1\}^n \to \{0,1\}^m$ can therefore be implemented as a unitary U_f.

There are in fact two ways to represent a classical function $f : \{0,1\}^n \to \{0,1\}^m$ as a unitary operation. The most general way (called *standard oracle* of f) is defined on computational basis states $|x\rangle|y\rangle$ (where $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$) by $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, where $\oplus$ corresponds to the bit-wise XOR operation. On the other hand, if $n = m$ and f is a permutation over $\{0,1\}^n$, then it is possible to represent f as a *minimal oracle* by $M_f|x\rangle = |f(x)\rangle$. There is in general no simple implementation of this minimal oracle since for a random permutation this requires to compute all possible outcomes in order to define the matrix representation of the unitary. Therefore this representation usually considered to be inefficient. This is further justified by the fact that this representation is in general more powerful than the standard oracle representation of classical functions as quantum unitaries. In particular, given access

[3] All unitaries can be rewritten up to a (unimportant) global phase as elements of $SU(2)$.

to the minimal oracle representation of a function, it is possible to solve some problems exponentially faster than using a standard oracle. Furthermore, constructing the minimal oracle using only calls to the standard oracle and its inverse requires at least $\mathcal{O}(n)$ calls to these oracles. Conversely, it is possible to construct the standard oracle with only one call to the minimal oracle and its inverse. See [74] for more information. However, there exists specific classical functions for which an efficient minimal oracle representation can be found (as shown later in this thesis, Section 5.3.1.2).

2.2.3 Useful Results from Quantum Information

We recall in this subsection a few principles and algorithms and operations that are useful in the rest of the thesis.

2.2.3.1 The Deutsch-Jozsa Algorithm

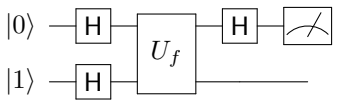
We recall here the principle of the Deutsch-Jozsa (or DJ) algorithm. The point of this algorithm is to solve the following promise problem: given a function f outputting a single bit, determine whether it is constant (the output bit is the same for all inputs) or balanced (half of the inputs output 0 and the other half output 1). The DJ algorithm solves this problem by using a single call to the standard oracle implementing the function f (with probability 1). It works in the following way (for a single bit of input):

Algorithm 1 Deutsch-Jozsa for Single Input Bit

Input: Oracle call U_f for binary function f.
Algorithm:

1. Prepare two qubits in the $|0\rangle|1\rangle$.
2. Apply a Hadamard gate to each of the two qubits.
3. Apply U_f with the second qubit receiving the output.
4. Apply a Hadamard gate to the first qubit.
5. Measure the first qubit in the computational basis and output the result.

This is represented as the following circuit:



A simple calculation gives that the state right before the application of the last Hadamard on the first qubit is (with $b_i = f(i)$ for $i \in \{0,1\}$ in the case of DJ for one input qubit):

$$
\begin{aligned}
|\phi\rangle =& U_f |+\rangle |-\rangle \\
=& U_f |0\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} + U_f |1\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
=& |0\rangle \frac{|f(0)\rangle - \left|\overline{f(0)}\right\rangle}{\sqrt{2}} + |1\rangle \frac{|f(1)\rangle - \left|\overline{f(1)}\right\rangle}{\sqrt{2}} \\
=& (-1)^{b_0} |0\rangle |-\rangle + (-1)^{b_1} |1\rangle |-\rangle \\
=& \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^{b_0 \oplus b_1} |1\rangle\right) \otimes |-\rangle
\end{aligned}
\tag{2.25}
$$

where $\bar{b} = 1 \oplus b$ is the binary complement of bit b and the last equality holds up to a global phase. This final state is equal to $|+\rangle |-\rangle$ if $b_0 \oplus b_1 = 0$ and $|-\rangle$ if $b_0 \oplus b_1 = 1$. It is therefore simple to see that after applying the Hadamard, the first qubit contains the classical value $b_0 \oplus b_1$, which is then the output value of the measurement. The same operations and calculations extend to the case with n input bits (and one output), which works by first generating a uniform superposition over the inputs by applying Hadamard gates to the $n+1$ bit of the state $|0\rangle^{\otimes n} |1\rangle$ and later applying Hadamard gates to the first n qubits after the call to the standard oracle U_f.

2.2.3.2 Quantum State Teleportation

We describe here the *teleportation of quantum states* which allows two parties A and B with pre-shared entanglement to send a qubit from on another by only sending two bits of classical information. A and B each hold one qubit of an EPR-pair. Suppose that A, in addition to half of the EPR-pair, also possesses a qubit in state $|\phi\rangle$ (not necessarily known to A) and wishes to send it to B. This can be done without quantum communication by applying the following procedure (with 1, 2 and 3 being the respective indices of the input qubit, A's half-EPR-pair and B's half-EPR-pair):

Protocol 1 Quantum Teleportation

Inputs: A has as input a quantum state $|\phi\rangle$. A and B have a pre-shared EPR-pair.
Protocol:

1. A applies a CNOT gate with qubit 1 as control and 2 as target.
2. A applies a Hadamard gate on qubit 1.
3. A measures both of its registers in the computational basis, let (b_1, b_2) be the outcomes. It sends these bits to B.
4. B applies $\mathsf{Z}^{b_1}\mathsf{X}^{b_2}$ to qubit 3.

If $|\phi\rangle = \alpha |0\rangle + \beta |1\rangle$, a simple calculation gives that before A measures its qubits, the joint state of the system is (with the convention that CNOT_{12} applies this gate with qubit 1 as control and 2 as target and H_1 applies a Hadamard on the first qubit):

$$\begin{aligned} |\Psi\rangle =& \mathsf{H}_1\mathsf{CNOT}_{12} |\phi\rangle \left|\psi^+\right\rangle \\ =& \frac{1}{\sqrt{2}}\mathsf{H}_1\mathsf{CNOT}_{12}(\alpha |0\rangle + \beta |1\rangle)(|00\rangle + |11\rangle) \\ =& \frac{1}{\sqrt{2}}\mathsf{H}_1(\alpha |000\rangle + \alpha |011\rangle + \beta |110\rangle + \beta |101\rangle) \end{aligned} \tag{2.26}$$

Finally, by applying the Hadamard gate on the first qubit and regrouping the appropriate terms, we get that the state is equal to:

$$\begin{aligned} \frac{1}{2}\Big[& |00\rangle (\alpha |0\rangle + \beta |1\rangle) + |01\rangle (\alpha |1\rangle + \beta |0\rangle) \\ & + |10\rangle (\alpha |0\rangle - \beta |1\rangle) + |11\rangle (\alpha |1\rangle - \beta |0\rangle)\Big] \end{aligned} \tag{2.27}$$

This can be rewritten as $\frac{1}{2}\sum_{b_1,b_2} |b_1 b_2\rangle \mathsf{Z}^{b_1}\mathsf{X}^{b_2} |\phi\rangle$. Then, if the measurement outcome of A is (b_1, b_2), the state in the quantum register of B is $\mathsf{X}^{b_2}\mathsf{Z}^{b_1} |\phi\rangle$. Applying the correction $\mathsf{Z}^{b_1}\mathsf{X}^{b_2}$ undoes the Quantum

One-Time-Pad and allows B to recover the state $|\phi\rangle$. Note that this means that before receiving the outcome bits from A, the state of B is perfectly mixed since it is equivalent to having received a Quantum One-Time-Padded state.

2.2.3.3 No-Cloning of Arbitrary Quantum States

The following theorem from [111] states that there exists no process that takes as input a quantum state and produces a perfect copy of it.

Theorem 2.3 (No-Cloning of Arbitrary States)**.** *There is no unitary operator U on Hilbert space $\mathcal{H} \otimes \mathcal{H}$ such that for all normalised states $|\phi\rangle$ and $|s\rangle$ in $\mathcal{H}$ we have:*

$$U |\phi\rangle |e\rangle = e^{i\alpha(\phi,e)} |\phi\rangle |\phi\rangle \tag{2.28}$$

for some real number α depending on ϕ and e.

Proof. Assume that there exist such a unitary U and let $|\psi\rangle$ and $|\phi\rangle$ be two pure states. Then:

$$\begin{aligned} U(|\psi\rangle \otimes |s\rangle) &= |\psi\rangle \otimes |\psi\rangle \\ U(|\phi\rangle \otimes |s\rangle) &= |\phi\rangle \otimes |\phi\rangle \end{aligned} \tag{2.29}$$

The inner product of these two state yields $\langle s| \langle\phi|U^\dagger U|\psi\rangle |s\rangle = \langle\phi|\psi\rangle$ for the left-hand side and $\langle\phi|\psi\rangle^2$ for the right hand side. This is possible if and only if $\langle\phi|\psi\rangle \in \{0, 1\}$, meaning that the states are either equal or orthogonal.

■

This theorem can be generalised to include CPTP-maps and mixed states via the technique of purification to transform the mixed state into a pure state and by using ancillary qubits to turn the CPTP-map into a unitary.

2.2.3.4 No-Communication through Local Operations

Quantum Information is a no-signalling theory, meaning that waves and in particular information carriers cannot travel faster than the speed of light and instant communication is impossible. It follows that no player can learn anything from a local operation performed by another player on a shared entangled state. With $\mathcal{E}_A$ being a CPTP-map on a register held by A, and shared state ρ_{AB} between players A and B, this can be expressed as:

$$\mathrm{Tr}_A(\rho_{AB}) = \mathrm{Tr}_A\left((\mathsf{I}_B \otimes \mathcal{E}_A)(\rho_{AB})\right) \tag{2.30}$$

2.2.3.5 State and Channel Pauli Twirl

A Pauli twirl occurs when a random Pauli operator is applied to a state or a channel. The result from the point of view of someone who does not know which Pauli has been used is a state or channel that is

averaged over all possible Pauli operators. The two following Pauli Twirling Lemmata then allow us to greatly simplify the analysis of the result of this averaging during our security proofs (for example for analysing the view of the Adversary after its attack on an encrypted state).

We start with the state Pauli twirl. This result is effectively a generalisation of Eq. 2.22 for the Q-OTP. An n-fold tensor of Pauli operators is defined as $P = \bigotimes_{j=1}^{n} \sigma_j$ where each σ_j is a single qubit Pauli operator $\{\mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$ or the identity I. Equivalently, since $\mathsf{Y} = i\mathsf{XZ}$, we can write P as $\mathsf{X}^a\mathsf{Z}^b$ for values $a, b \in \{0,1\}^n$, where $\mathsf{X}^a = \bigotimes_{i=1}^{n} \mathsf{X}^{a_j}$ and similarly for Z^b. Let $\mathcal{P}_n = \{\mathsf{X}^c\mathsf{Z}^d\}_{c,d\in\{0,1\}^n}$ be the set of all n-fold tensor products of Pauli operators and the identity.

Lemma 2.4 (State Pauli Twirling)**.** *Let ρ be a density matrix representing an n-qubit mixed state. Then, with $\mathsf{I}/2^n$ being the perfectly mixed state over n qubits:*

$$\frac{1}{2^{2n}} \sum_{P\in\mathcal{P}_n} P\rho P^\dagger = \frac{\mathsf{I}}{2^n} \tag{2.31}$$

Proof. We decompose $\rho = \sum_{k,j\in\{0,1\}^n} \alpha_{kj} |k\rangle\langle j|$ into its basis components. We use the $\cdot$ operator here as a the scalar product of two binary vectors, i.e. $x \cdot y = \sum_{i\in[n]} x_i y_i$ for $x, y \in \{0,1\}^n$. Then for each element $|k\rangle\langle j|$ we have:

$$\begin{aligned} \frac{1}{2^{2n}} \sum_{P\in\mathcal{P}_n} P |k\rangle\langle j| P^\dagger &= \frac{1}{2^{2n}} \sum_{c,d\in\{0,1\}^n} \mathsf{X}^c\mathsf{Z}^d |k\rangle\langle j| \mathsf{Z}^d\mathsf{X}^c \\ &= \frac{1}{2^{2n}} \sum_{c,d\in\{0,1\}^n} (-1)^{d\cdot(k\oplus j)} |k\oplus c\rangle\langle j\oplus c| \end{aligned} \tag{2.32}$$

The second equality comes from the fact that applying an X operation is equivalent to a classical bit flip and Z adds a phase of -1 if the qubit is in the state $|1\rangle$. Then we use the fact that $\sum_{d\in\{0,1\}^n} (-1)^{d\cdot(k\oplus j)} = 0$ if $k \neq j$ and it is equal to 1 if $k = j$. Therefore, continuing from the previous result, we get:

$$\frac{1}{2^n} \sum_{c\in\{0,1\}^n} |j\oplus c\rangle\langle j\oplus c| = \frac{\mathsf{I}}{2^n} \tag{2.33}$$

Then, for the general state ρ:

$$\frac{1}{2^{2n}} \sum_{k,j\in\{0,1\}^n} \alpha_{kj} \sum_{P\in\mathcal{P}_n} P |k\rangle\langle j| P^\dagger = \sum_{j\in\{0,1\}^n} \alpha_{jj} \frac{\mathsf{I}}{2^n} = \mathrm{Tr}(\rho)\frac{\mathsf{I}}{2^n} = \frac{\mathsf{I}}{2^n} \tag{2.34}$$

■

We will then require the following Pauli Twirling Lemma from [34] for some of the proofs in this thesis.

Lemma 2.5 (Channel Pauli Twirling)**.** *Let ρ be a density matrix representing an n-qubit state and Q, Q' two n qubit Paulis from $\mathcal{P}_n$. Then, if $Q \neq Q'$, we have:*

$$\sum_{P\in\mathcal{P}_n} P^\dagger Q P \rho P^\dagger Q'^\dagger P = 0 \tag{2.35}$$

Proof. We use the fact that $Q = \mathsf{X}^a\mathsf{Z}^b$, $Q' = \mathsf{X}^{a'}\mathsf{Z}^{b'}$ for some value of $a, b, a', b' \in \{0,1\}^n$. The same sum rewrite as in the previous proof gives us:

$$\sum_{P\in\mathcal{P}_n} P^\dagger Q P \rho P^\dagger Q'^\dagger P = \sum_{c,d\in\{0,1\}^n} \mathsf{Z}^d\mathsf{X}^c\mathsf{X}^a\mathsf{Z}^b\mathsf{X}^c\mathsf{Z}^d\rho\mathsf{Z}^d\mathsf{X}^c\mathsf{Z}^{b'}\mathsf{X}^{a'}\mathsf{X}^c\mathsf{Z}^d \tag{2.36}$$

We then use the fact that $\mathsf{X}^2 = \mathsf{Z}^2 = \mathsf{I}$ along with the commutation relation $\mathsf{XZ} = -\mathsf{ZX}$, which in case of controlled operations by bit strings yields $\mathsf{X}^x\mathsf{Z}^z = (-1)^{x\cdot z}\mathsf{XZ}$ with the same convention for the $\cdot$ scalar product as in the previous proof:

$$\ldots = \sum_{c,d\in\{0,1\}^n} (-1)^{c\cdot(b\oplus b')+d\cdot(a\oplus a')}\mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^{b'}\mathsf{X}^{a'} = \mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^{b'}\mathsf{X}^{a'}\sum_{c\in\{0,1\}^n}(-1)^{c\cdot(b\oplus b')}\sum_{d\in\{0,1\}^n}(-1)^{d\cdot(a\oplus a')} \tag{2.37}$$

Finally, since $Q \neq Q'$, either $a \neq a'$ or $b \neq b'$, meaning that either $\sum_{c\in\{0,1\}^n}(-1)^{c\cdot(b\oplus b')} = 0$ or $\sum_{d\in\{0,1\}^n}(-1)^{d\cdot(a\oplus a')} = 0$, concluding the proof.

■

2.3 Measurement-Based Quantum Computing

This section provides a brief overview of various useful notions linked to another model of quantum computation, namely Measurement-Based Quantum Computing (MBQC). Based on the gate-teleportation principle, it was shown in [118] that it can implement universal quantum computing and therefore that the MBQC model has the same power as the circuit model presented above. The correspondence between one and the other is described with the tools of measurement calculus [36]. MBQC works by choosing an appropriate entangled state and then by measuring single qubits and, depending on the outcomes, applying correction operators to the rest. It is therefore a natural setup for considering delegated computations, i.e. when a Client with limited quantum capabilities only has to provide its quantum inputs and instruct a more powerful Server to perform a computation on its behalf, while the Server takes on the responsibility of creating the large entangled state.

The basic MBQC Protocol for classical inputs and outputs can be summarized as follows. Any computation chosen by the Client is first translated into a graph $G = (V, E)$, where two vertices sets I and O define input and output vertices, and a list of angles $\{\phi(v)\}_{v\in O^c}$. While the discussions below hold for angles in $[0, 2\pi)$, if we settle for approximate universality it is sufficient to restrict ourselves to the set of angles $\Theta = \{k\pi/4\}_{k\in\{0,1,\ldots,7\}}$ (see [21]). The set $\{G, I, O, \{\phi(v)\}_{v\in O^c}\}$ is called a measurement pattern. To run a computation, the Client instructs the Server to prepare the *graph state* $|G\rangle$: for each vertex in V, the Server creates a qubit in the state $|+\rangle$ and performs a CZ gate for each pair of qubits forming an edge in G. The Client then asks the Server to measure each qubit of O^c along the basis $\left\{\left|+_{\phi'(v)}\right\rangle\!\left\langle+_{\phi'(v)}\right|, \left|-_{\phi'(v)}\right\rangle\!\left\langle-_{\phi'(v)}\right|\right\}$ for updated angle $\phi'(v)$ in the order defined by the *flow of the computation* defined below. The Server then returns the measurement result $s(v)$. After measuring all qubits in O^c according to these angles, the Server returns all output qubits $v \in O$ to the Client on which the Client performs the final Pauli correction $\mathsf{Z}^{s^Z(v)}\mathsf{X}^{s^X(v)}$. We now define how these corrections are calculated.

Since the computation is performed in MBQC by measuring parts of a large entangled state and these measurements are probabilistic, we must guarantee that the result of the computation is the same for each possible branch of measurement results, i.e. if we take the standard measurement outcome to be 0 for all non-output qubits, it is possible to correct the effect of obtaining the measurement outcome 1 on any such qubit. To find how to correct the measurement outcome, we use the fact that any graph state $|G\rangle$ with input space I is an *eigen-state* of the following set of operators (called *stabilisers*) for all non-input vertices $v \in I^c$:

$$K_v := \mathsf{X}_v \bigotimes_{w \in N_G(v)} \mathsf{Z}_w \tag{2.38}$$

We then have $K_v |G\rangle = |G\rangle$, meaning that applying any operator K_v does not modify the graph state. If the measurement outcome is 1 for a given vertex $v \in O^c$, then this is equivalent to having first applied Z_v before the measurement and then obtaining the measurement outcome 0. For the effect of this measurement outcome to be correctable, it must be possible to apply a subset of the graph-state stabilisers in such a way that the combined effect on the state both compensates this Z_v operation and applies Pauli operations on qubits that have not yet been measured. This is possible if the graph has an associated flow of computation.

The flow consists of a function $f : O^c \to I^c$ from measured qubits to non-input qubits and a partial order ($\preceq$) over the vertices of the graph. They must furthermore satisfy the following conditions.

Definition 2.3 (Flow of Computation, Taken from [35])**.** *A graph $G = (V, E, I, O)$ is said to have a flow if there exists a map $f : O^c \to I^c$ and a partial order ($\preceq$) over V such that for all $v \in O^c$:*

1. *$(v, f(v)) \in E$;*
2. *$v \preceq f(v)$;*
3. *for all neighbours $\tilde{v} \in N_G(v)$, $f(v) \preceq \tilde{v}$.*

The existence of such a flow in a graph used for computations in MBQC patterns then guarantees that the computation can be performed deterministically by performing measurements on qubits in the order given by $\preceq$ [35], meaning that the unitary applied is independent from the results of the measurements on the non-output qubits. It is a sufficient but non-necessary condition [22] but all graphs used in this thesis will satisfy it. In an MBQC computation, each qubit v is said to be X-dependent on $X(v) = f^{-1}(v)$ and Z-dependent on all qubits $\tilde{v}$ such that $v \in N_G(\tilde{v})$ (this set is called $Z(v)$). We also define the past of qubit v, which includes all the qubits upon which qubit v is dependent.

Definition 2.4 (Past of qubit v and Influence-past of qubit v)**.** *We define the past of qubit v as $Past(v) = Z(v) \cup X(v)$ to be the set of qubits $\tilde{v}$ that have X or Z dependency on v. We define the set of influence-pasts of qubit v as $\{c(v)\} = \{0,1\}^{\#Past(v)}$, where each $c(v)$ corresponds to the string of measurement outcomes $s_{\tilde{v}} \in \{0,1\}$ for all qubits $\tilde{v} \in Past(v)$.*

The sets $X(v)$ and $Z(v)$ contain the vertices whose measurement outcomes may imply a Pauli correction on vertex v. However it is possible to refrain from applying actual Pauli operations on the vertex v and instead modify the measurement angle with which this vertex is measured. This can be done using the fact that $\mathsf{X} |+_\theta\rangle = |+_{-\theta}\rangle$ and $\mathsf{Z} |+_\theta\rangle = |+_{\theta+\pi}\rangle$ and leads to the definition of the flow-updated

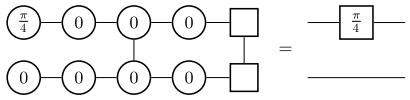
measurement angle $\phi'(v)$. Given the sets $X(v)$ and $Z(v)$, the computation angle for qubit v needs to be adjusted as such: let $s^X(v) = \bigoplus_{\tilde{v} \in D^X(v)} s(\tilde{v})$ and $s^Z(v) = \bigoplus_{\tilde{v} \in D^Z(v)} s(\tilde{v})$, where $s(v)$ corresponds to the outcome of the measurement on qubit v and $D^X(v)$ and $D^Z(v)$ are subsets of $X(v)$ and $Z(v)$ respectively (these qubits in $D^X(v)$ and $D^Z(v)$ have all already been measured as they belong to the past neighbours and past neighbours of past neighbours, see the flow construction in [35]). Then the corrected angle (the one that is actually measured) is (details can be found in [68]):

$$\phi'(v) = (-1)^{s^X(v)}\phi(v) + s^Z(v)\pi \tag{2.39}$$

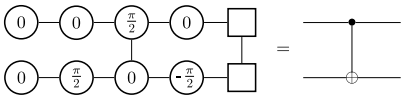
Figure 2.1 presents diagrams (taken from [21]) showing how to translate a universal set of gates to the MBQC model using the brickwork graphs.

(a) Implementation of identity gate.

(b) Implementation of Hadamard gate.



(c) Implementation of $\mathsf{Z}(\pi/4)$ gate.



(d) Implementation of CNOT gate.

Figure 2.1: Translation of identity, H, $\pi/4$ and CNOT gates. Each vertex represents a qubit in the state $|+\rangle$, each edge corresponds to a CZ operation between connected vertices and each value is the default measurement angle (before teleportation corrections are taken into account). Taken from [21].

2.3.1 Graph State Bridge Operation

We now describe the basic process of bridge operations on graph states, introduced in [68], which will be used in our MPQC Protocol later. The input to this operation is of the following form $\mathsf{CZ}_{1,2}\mathsf{CZ}_{2,3}(\rho \otimes |+\rangle \otimes \sigma)$, where ρ and σ are arbitrary qubit mixed states, the qubits are respectively indexed 1, 2 and 3, and the operation $\mathsf{CZ}_{v,w}$ applies a CZ operation to qubits v and w. The purpose of the bridge operation is to delete the middle qubit 2 along with its corresponding edges and join qubits 1 and 3 by a new edge. The description of this operation is given by the following Protocol 2.

Protocol 2 Bridge Operation on Three-Qubit Line Graph

Measure qubit 2 in the basis $\{|+_{\pi/2}\rangle, |-_{\pi/2}\rangle\}$ and record measurement result $b \in \{0, 1\}$.
Apply operations $\mathsf{Z}(-\pi/2)_1$ and $\mathsf{Z}(-\pi/2)_3$, where 1 and 3 are the remaining qubits.
Apply operations Z_1^b and Z_3^b.

Lemma 2.6 shows that performing the bridge operation on the three qubit line graph is equivalent to constructing a line graph with only qubits 1 and 3.

Lemma 2.6 (Correctness of Bridge Operation)**.** *The output of Protocol 2 after tracing out the second subsystem is* $\mathsf{CZ}_{1,3}(\rho \otimes \sigma)$ *where the qubits are respectively numbered* 1 *and* 3*.*

Proof. It is sufficient to prove the lemma in the case where ρ is a pure state of the form $\alpha |0\rangle + \beta |1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$. Similarly, because the following holds for any pure state σ, the lemma holds for any σ. The original line graph state is given by:

$$\frac{1}{\sqrt{2}}(\alpha |0\rangle (|0\rangle \sigma + |1\rangle Z\sigma) + \beta |1\rangle (|0\rangle \sigma - |1\rangle Z\sigma) \tag{2.40}$$

If result of the measurement on the middle qubit is $b = 0$, then the remaining state is (tracing out the system containing the second qubit):

$$\frac{1}{\sqrt{2}}(\alpha |0\rangle \sigma - i\alpha |0\rangle Z\sigma + \beta |1\rangle \sigma + i\beta |1\rangle Z\sigma) \tag{2.41}$$

which becomes, after the $\mathsf{Z}(-\pi/2)_1$ and $\mathsf{Z}(-\pi/2)_3$ operations:

$$\alpha |0\rangle \frac{1-i}{\sqrt{2}}\sigma + \beta |1\rangle \frac{1-i}{\sqrt{2}} Z\sigma \tag{2.42}$$

which corresponds to the state $\mathsf{CZ}_{1,3}(\rho \otimes \sigma)$ up to a global phase $\frac{1-i}{\sqrt{2}}$. If the result of the measurement on the middle qubit is $b = 1$, the remaining state is:

$$\frac{1}{\sqrt{2}}(\alpha |0\rangle \sigma + i\alpha |0\rangle Z\sigma + \beta |1\rangle \sigma - i\beta |1\rangle Z\sigma) \tag{2.43}$$

which becomes, after the $\mathsf{Z}(-\pi/2)_1$ and $\mathsf{Z}(-\pi/2)_3$ operations:

$$\alpha |0\rangle \frac{1+i}{\sqrt{2}} Z\sigma - \beta |1\rangle \frac{1+i}{\sqrt{2}}\sigma \tag{2.44}$$

which, after the Z_1 and Z_3 operations becomes:

$$\alpha |0\rangle \frac{1+i}{\sqrt{2}}\sigma + \beta |1\rangle \frac{1+i}{\sqrt{2}} Z\sigma \tag{2.45}$$

which corresponds to the state $\mathsf{CZ}_{1,3}(\rho \otimes \sigma)$ up to global phase $\frac{1+i}{\sqrt{2}}$.

■

Note that, because the operations applied in Protocol 2 commute with diagonal operations, all qubits v can be pre-rotated using operations $\mathsf{Z}(\theta(v))$ for arbitrary angles $\theta(v)$. This means that it can also be used in the secure protocols extended from MBQC found in Sections 3.5.1 and 3.5.2.

Chapter 3

Cryptographic Security Frameworks

Definitions and results from classical and quantum cryptography are described in this chapter. They are then used in the rest of the thesis and each subsequent chapter recalls the exact applicable framework of security.

We start by introducing some basic cryptographic notations (the quantum states corresponding to the special messages are orthogonal among themselves and to any computational basis states):

- the messages Abort and Ok signal respectively that a party aborted or completed the protocol correctly;
- Ack is sent by a party as acknowledgement that it has completed a step in the protocol;
- End is sent by a party terminating a specific task;
- Corrupted is sent if a party notices malicious behaviour from a specific player;
- the dummy input λ is used by players that do not have a proper honest input to a given protocol to signal that they are ready to begin;
- the security parameter η is passed implicitly as 1^η to all parties, Adversaries and Distinguishers;
- a function f is polynomial in η, written $f(\eta) = poly(\eta)$, if there exists a constant c such that $f(\eta) = \mathcal{O}(\eta^c)$.
- a function ϵ is *negligible in* η if, for every polynomial p, for η sufficiently large it holds that $\epsilon(\eta) < 1/p(\eta)$;
- a function $\mu(\eta)$ is *overwhelming* if there exists a negligible $\epsilon(\eta)$ such that $\mu(\eta) = 1 - \epsilon(\eta)$;

3.1 Basic Cryptographic Primitives

We start by describing the basic building blocks that will be used in the rest of this thesis for protocol construction. We start by an informal presentation of various primitives (the formal presentations of which are given later in simulation-based frameworks in Section 3.3.3) and later focus on Bit Commitment,

for which we use the stand-alone security criteria (as opposed to its definition as an Ideal Functionality as in fully composable frameworks of security).

3.1.1 Common Primitives

Communication Channels. In the course of this thesis, we use common resources such as the Authenticated Classical Channel, the Confidential Classical Channel, the Secure Classical Channel, the Insecure Quantum Channel and the Confidential Quantum Channel. The difference between the classical channels is that the first one guarantees that a message originating from a party has not been modified but anyone can read the contents, while the second one provides the opposite functionality, meaning that no one apart from the Sender and Receiver can read the message but it may be modified at will without the honest parties noticing. The third one combines the best of both worlds by enforcing both the secrecy and non-malleability of the message. The Private Quantum Channel is the quantum equivalent of the Confidential Classical Channel: the message is hidden but can be tampered with.

Common Reference String. The Common Reference String is a common trusted setup assumption, providing two players with the same starting information produced using a known efficient algorithm. This is preferred over having one player produce this string and send it to the other in cases where this player may benefit from either producing an incorrect string, or if the process of producing this information necessarily gives the player generating it too much information as a by-product.[1]

Key-Distribution The purpose of key-distribution is to sample uniformly at random a binary key and sends the same to two parties, while an Eavesdropper can only recover the length of the key. This can be extended to multiple players and the Eavesdropper's leakage might be different for some protocols (for example, an upper-bound on the key size). Other distributions for the key may also be considered but this is sufficient in our case.

Coin-Tossing. The Two-Party Coin-Tossing (Resource 8) has no input and distributes to both parties an output sampled from a specified distribution which must be efficiently samplable.[2] We suppose that P_1 receives the output first and decides then whether to continue or not.

1-out-of-2 Oblivious Transfer. A 1-out-of-2 Oblivious Transfer is a two-party functionality in which one party inputs two strings (x_0, x_1) and the other (P_2) inputs a bit $b \in \{0, 1\}$. At the end of the protocol P_2 recovers x_b. On one hand P_1 should not know which of the strings P_2 has chosen while P_2 has no information about the string x_{1-b} that it did not choose.

Classical Secure Multi-Party Computation. It allows N Clients to provide their private inputs and perform a collectively defined computation C on them with the guarantee that the computation is performed properly. We assume that it keeps an internal state between calls. Since it is impossible to

[1] For example the string might be a public key for an encryption scheme that both players need in order to perform the protocol, but the security of the protocol rests on the fact that neither player knows the corresponding secret key.

[2] A distribution is said to be efficiently samplable if a sample from the distribution can be produced on an empty input by a PPT machine.

guarantee fairness in output distribution with a dishonest majority, some participants may receive their output before others (and choose to abort, preventing the rest of the participants from obtaining theirs).

Blind Delegated Quantum Computation. The primitive allows a single Client to run a quantum computation on a Server so that the Server doesn't learn anything of the computation and input besides some predefined leakage l_ρ. The Server can however make the Client accept an incorrect computation.

The next primitives correspond to those that our protocols will strive to construct.

Verifiable Blind Delegated Quantum Computation. This is a strengthening of the primitive above in the sense that it allows a single Client to run a quantum computation on a Server so that the Server doesn't learn anything besides the leakage l_ρ but also cannot corrupt the computation.

Classical Input Two-Party Quantum Computation. This functionality implements Two-Party Quantum Computation for unitary U with classical inputs and quantum outputs. It takes as input two strings (one from each party), creates the associated quantum states and applies unitary U to the states and a predetermined number of ancillae, after which it sends the output registers to each player. We suppose here that player P_1 recovers the output first.

Multi-Party Quantum Computation. Its purpose is to allow N Clients to perform a collectively defined computation U over their private quantum inputs with the guarantee that their computation is either executed properly or it is aborted altogether. Without loss of generality, we suppose that each Client has a single qubit of input. We suppose that the primitive may leak a value l_ρ about the parties' intended computation and input to an Eavesdropper. As above, it is impossible to guarantee fairness of output distribution in the case of a dishonest majority and therefore the malicious parties can always choose to receive their output before the honest players.

3.1.2 Classical Bit Commitment

Bit Commitment consists of two phases, Commit and Reveal, such that after the Commit phase the Receiver has no information about the value that has been committed (hiding), while during the Reveal phase the Sender cannot reveal a value different from the one used previously to create the commitment (binding).

More formally, let $(\mathsf{Com}, \mathsf{Verif})$ be a pair of classical polynomial-time algorithms. Com (a probabilistic algorithm) takes as input the committed message $m \in \mathfrak{M}$ (from a given message space $\mathfrak{M}$), and outputs (c, u), where c corresponds to the commitment and u to the opening information. Verif is deterministic and takes as input (m, c, u) and outputs a bit $b \in \{0, 1\}$ (with $b = 1$ indicating that the commitment has been correctly verified).

As a first requirement, the commitment scheme is said to have *perfect completeness* if all commitments generated by Com are verified as valid by Verif:

Definition 3.1 (Perfect Completeness)**.** *The commitment scheme* $(\mathsf{Com}, \mathsf{Verif})$ *is* perfectly complete *if for all* $m \in \mathfrak{M}$ *and* $(c, u) \leftarrow \mathsf{Com}(m)$*, we have* $\mathsf{Verif}(c, m, u) = 1$*.*

To guarantee the Sender's security, the commitment by itself should not reveal any information about the message being committed, which is captured by the following definition (for the computational version, the Adversary is quantified over all QPT Adversaries, with auxiliary input given by a QPT Environment):

Definition 3.2 (Statistically Hiding Commitment)**.** *The commitment scheme is said to be* statistically hiding *if for all Adversaries $\mathcal{A}$ and all polynomial l, there exists a ϵ_H negligible in η such that for all message pairs (m_0, m_1) with $\#m_i \leq l(\eta)$, for all auxiliary input of the Adversary $\rho_{\mathcal{A}}$, $|P_0 - P_1| \leq \epsilon_H(\eta)$ with $P_i = \Pr[b = 1 \mid (c, u) \leftarrow \mathsf{Com}(m_i), b \leftarrow \mathcal{A}(\rho_{\mathcal{A}}, c)]$. We say that a commitment scheme is* perfectly hiding *if $\epsilon_H = 0$.*

Conversely, to protect the Receiver, we require as well that the Bit Commitment primitive is *collapse-binding* [129]. Essentially, it captures the fact that, if a commitment has been sent, a computationally-bounded Adversary is not able to distinguish whether a quantum register containing the corresponding committed message has been measured in the computational basis of not (meaning that the state of this register was close to one which had already been measured). We start by defining this property for general relations R on sets $\mathfrak{X}$ and $\mathfrak{Y}$. This definition is adapted from [38].

Definition 3.3 (Collapsing Relation)**.** *Let $\mathfrak{X}$ and $\mathfrak{Y}$ be two sets and $R \subset \mathfrak{X} \times \mathfrak{Y}$ a relation on those sets. Let $\mathcal{X}$ and $\mathcal{Y}$ be two quantum registers and let $\mathbb{M}(\cdot)$ denote a measurement of a quantum register in the computational basis. Consider the following games:*

$$\begin{array}{lllll} \textit{Game1}: & (\mathcal{X}, \mathcal{Y}) \leftarrow \mathcal{A}; & x \leftarrow \mathbb{M}(\mathcal{X}); & y \leftarrow \mathbb{M}(\mathcal{Y}); & b \leftarrow \mathcal{B}(\mathcal{X}, \mathcal{Y}) \\ \textit{Game2}: & (\mathcal{X}, \mathcal{Y}) \leftarrow \mathcal{A}; & x \leftarrow \mathbb{M}(\mathcal{X}); & & b \leftarrow \mathcal{B}(\mathcal{X}, \mathcal{Y}) \end{array}$$

A QPT *Adversary $(\mathcal{A}, \mathcal{B})$ is called* valid *if it only outputs values that satisfy the relation:*

$$\Pr[R(x, y) = 1 \mid (x, y) \leftarrow \mathbb{M}(\mathcal{X}\mathcal{Y})] = 1 \tag{3.1}$$

R is said to be y-collapsing given x *if there exists a ϵ_c negligible in η such that, for all valid* QPT *Adversaries $(\mathcal{A}, \mathcal{B})$:*

$$|\Pr[b = 1 \mid \textit{Game1}] - \Pr[b = 1 \mid \textit{Game2}]| \leq \epsilon_c(\eta) \tag{3.2}$$

We then apply this notion to commitments, yielding the following definition:

Definition 3.4 (Collapse-Binding Commitment)**.** *Let $(\mathsf{Com}, \mathsf{Verif})$ be a commitment scheme with message, commitment and opening sets $(\mathfrak{M}, \mathfrak{C}, \mathfrak{U})$. The commitment scheme is said to be* collapse-binding *if the relation* Verif *is m-collapsing given c, where m is a message and c a commitment.*

The explicit version of the definition is given as well:

Definition 3.5 (Collapse-Binding Commitment, Explicit)**.** *Let $(\mathsf{Com}, \mathsf{Verif})$ be a commitment scheme and let $(\mathcal{S}, \mathcal{M}, \mathcal{U})$ be quantum registers. Let $\mathbb{M}(\cdot)$ denote a measurement of a quantum register in the computational basis. Consider the following games:*

$$\begin{array}{lllll} \textit{Game1}: & (\mathcal{S}, \mathcal{M}, \mathcal{U}, c) \leftarrow \mathcal{A}; & m \leftarrow \mathbb{M}(\mathcal{M}); & b \leftarrow \mathcal{B}(\mathcal{S}, \mathcal{M}, \mathcal{U}, c) \\ \textit{Game2}: & (\mathcal{S}, \mathcal{M}, \mathcal{U}, c) \leftarrow \mathcal{A}; & & b \leftarrow \mathcal{B}(\mathcal{S}, \mathcal{M}, \mathcal{U}, c) \end{array}$$

A QPT *Adversary* $(\mathcal{A}, \mathcal{B})$ *is called* valid *if it only outputs commitments passing verification:*

$$\Pr[\mathsf{Verif}(c, m, u) = 1 \mid m \leftarrow \mathbb{M}(\mathcal{M}), u \leftarrow \mathbb{M}(\mathcal{U})] = 1 \tag{3.3}$$

A commitment scheme $(\mathsf{Com}, \mathsf{Verif})$ *is said to be* collapse-binding *if there exists a* μ *negligible in* η *such that, for all valid Adversaries* $(\mathcal{A}, \mathcal{B})$*:*

$$|\Pr[b = 1 \mid \mathit{Game1}] - \Pr[b = 1 \mid \mathit{Game2}]| \leq \mu(\eta) \tag{3.4}$$

This definition means that, if the commitment has been measured and the quantum registers contain only messages and opening informations that are valid, no computationally-bounded Adversary can distinguish whether the message register has been measured or not. It is a quantum variant of classical computationally-binding commitments.[3] The fact that a bounded Adversary should not notice that the register containing the message has been measured implies that the Adversary cannot open the commitment to two different values with more than negligible probability.[4]

3.2 Model for Quantum Networked Machines

We present here the definitions and notations that will be common to all security models in this thesis unless otherwise specified.

Efficient classical algorithms are defined as probabilistic polynomial-time, or PPT, Turing machines (i.e. able to solve problems from the complexity class BPP). They are able to perform any computations provided that their running time is upper-bounded by a fixed polynomial in the input length. Thay may furthermore use a polynomial number of uniformly random coin-flips. On the other hand, all efficient quantum parties are considered to be Quantum Polynomial Time, or QPT, machines (the quantum equivalent of PPT), which are also called polynomial-time quantum Turing machines and recognise languages in the BQP class of complexity [27, 107]. They can perform any polynomial-sized family of quantum circuits and interact quantumly with other participants (by sending quantum states which may or may not be in superposition).

The formal definition of complexity class BQP (and by extension of efficient quantum machines) is given in Definition 3.6 [27]. This is defined in terms of languages efficiently accepted by quantum machines, or equivalently the class of decision problems efficiently solvable by quantum computers. It is by extension used to describe any efficient quantum computation, regardless of whether it solves a problem in the complexity-theoretic sense.

[3] A commitment scheme is classical computationally-binding if any computationally-limited Adversary is capable of opening the commitment to two different values of its choice at most with negligible probability. Theorem 22 from [129] shows that this property is insufficient for guaranteeing the security of protocols in the quantum setting. The idea is that it is possible to construct a state $|\psi\rangle$ and two efficient unitaries U_0 and U_1 such that applying U_0 to $|\psi\rangle$ and measuring it in the computational basis yields one valid opening for a commitment, while doing this with U_1 would yield a different opening (both chosen by the Adversary). Having such a state would allow an Adversary with auxiliary input $|\psi\rangle$ to cheat in protocols that use commitments. However it does not allow it to produce two valid openings *at the same time* since it has only one copy of $|\psi\rangle$, which cannot be reused after producing one opening, and therefore it cannot break the classical binding property.

[4] Otherwise it could simply produce these messages, store them in superposition in the quantum register and easily test whether the register has been measured on the positions where the two messages differ. This is not proven in [129] but a formalised version would follow directly from the proof of Lemma 25 in that paper.

Definition 3.6 (Languages in BQP)**.** *We say that a language L is in* BQP *if and only if there exists a polynomial-time uniform family of quantum circuits* [5] $\{Q_n \mid n \in \mathbb{N}\}$ *such that:*

- *Q_n takes n qubits as input and outputs a single classical bit.*
- *For all $x \in L$, $\mathbb{P}\Big[b = 1 \mid b \leftarrow Q_{\#x}(x)\Big] \geq 2/3$.*
- *For all $x \notin L$, $\mathbb{P}\Big[b = 0 \mid b \leftarrow Q_{\#x}(x)\Big] \geq 2/3$.*

The parties participating in a given protocol consist of a number of players, which are supposed to be efficient (quantum or classical), an Adversary which may control any fixed number of parties chosen at the beginning of the protocol (we call such adversaries *static*), and an Environment which represents intuitively "anything that happens outside of the protocol's execution". We often abuse notation and consider a corrupted party and the Adversary as one entity. The Adversary and the Environment will always have quantum capabilities but can be either quantum computationally-bounded (leading to computational security) or unbounded (yielding unconditional or statistical security). The Adversary can furthermore be either Quantum Honest-but-Curious (also called specious [44]), where it can always produce if asked an honest state that is coherent with the transcript, or fully Malicious, a setting where the actions of the Adversary are only constrained by its own computational power.

Each party has access to quantum registers $(\mathcal{C}_i, \mathcal{Q}_i, \mathcal{W}_i)$ which correspond respectively to a classical communication register, a quantum communication register and an internal work (or state) register. $\mathcal{C}_i$ and $\mathcal{Q}_i$ are initialised to $\left|0^{\#\mathcal{C}_i+\#\mathcal{Q}_i}\right\rangle$ while the work register is initialised with the party's input. The Adversary has access to an additional register $\mathcal{W}_\mathcal{A}$, an internal work (or state) register that is initialised with the Adversary's input. At each step of the protocol where party P_i is activated, first its classical and quantum communication registers $\mathcal{C}_i$ and $\mathcal{Q}_i$ are initialised with the classical and quantum communications that it receives at this round, then a unitary transformation that depends on the protocol round is applied to all three registers, and finally $\mathcal{C}_i$ is measured in the computational basis and the outcome is recorded. Note that an alternative acceptable formulation is to have a family of possible unitaries indexed by the classical messages, or equivalently unitaries controlled by the classical messages.

The Environment $\mathcal{Z}$ produces (and initialises the corresponding registers):

- An input state ρ_i for each party i (honest or corrupted). These inputs may be quantum or classical depending on the functionality.
- An input state $\rho_\mathcal{A}$ for the Adversary.
- An additional register $\mathcal{W}_\mathcal{Z}$, which it keeps to itself and is used to store information about the inputs.
- This joint input is denoted $\rho_{in} \in D(\bigotimes_i \mathcal{W}_i \otimes \mathcal{W}_\mathcal{A} \otimes \mathcal{W}_\mathcal{Z})$.

Alternatively, in the case of two parties (but this is easily generalisable to a larger number of participants), an n-round protocol between players A and B with internal registers collectively denoted $\mathcal{X}$ and $\mathcal{Y}$ respectively, along with communication register $\mathcal{C}$, can be seen as a succession of CPTP-maps $\mathcal{E}_1 \circ \mathcal{F}_1 \circ \ldots \circ \mathcal{E}_n \circ \mathcal{F}_n$ where $\mathcal{E}_i : L(\mathcal{A} \otimes \mathcal{C}) \to L(\mathcal{A} \otimes \mathcal{C})$ and $\mathcal{F}_i : L(\mathcal{B} \otimes \mathcal{C}) \to L(\mathcal{B} \otimes \mathcal{C})$ applied to some initial state ρ_{in} (without loss of generality, the communication register is initialised to $|0\rangle^{\otimes \#\mathcal{C}}$).

[5] There exists a polynomial-time deterministic Turing machine taking as input 1^n for $n \in \mathbb{N}$ and outputting a classical description of Q_n.

3.3 "Ideal vs. Real" Frameworks of Security

3.3.1 Stand-Alone Model of Security

The two security frameworks used in this thesis follow the *ideal/real simulation paradigm*, with the first one being based on the Stand-Alone Model of [66, 127]. A protocol is considered as secure if it is a good approximation of an ideal version called Ideal Functionality.

From the real-world Adversary $\mathcal{A}$ controlling a corrupted party, the proof of security constructs an ideal-world Adversary (also called Simulator) $\mathcal{S}$ that attacks the ideal execution of the functionality that the protocol emulates, represented by a trusted third party. This Simulator runs the Adversary $\mathcal{A}$ internally by applying at each step of the protocol a black-box unitary transformation to its three quantum registers that depends on the classical message that the Adversary receives in this step (this unitary corresponds to the quantum circuit of the Adversary or its inverse). After each such activation, the Simulator measures the classical message register of the Adversary in the computational basis. The Simulator also has single-query access to an oracle which implements the Ideal Functionality (meaning that it can send inputs and recover outputs corresponding to the corrupted player but no more).

At the end of the execution, the Environment receives a state from the Adversary along with the outputs of all honest parties and performs computations on them and its additional register $\mathcal{W}_{\mathcal{Z}}$, outputting a single bit. The Simulator must behave so that the Environment $\mathcal{Z}$, based solely on the inputs and outputs of the participants and the Adversary, is not able to detect that it is not in fact interacting directly with real world parties instead (this guess is represented by the output bit of the Environment). The standard definition of computational security is therefore satisfied if the simulated and real states are indistinguishable for the Environment $\mathcal{Z}$. We present the definition of security in the two-party case.

Definition 3.7 (Computational Security against Malicious Adversaries)**.** *A protocol* Π *$\epsilon(\eta)$-securely emulates Ideal Functionality* $\mathcal{F}$ *against computationally-bounded A if there exists a* QPT *Simulator* $\mathcal{S}_{\mathcal{A}}$ *such that for all* QPT *Malicious Adversaries* $\mathcal{A}$ *controlling the corrupted party* A^* *and all* QPT *Environments* $\mathcal{Z}$*:*

$$\left|\Pr\left[1 \leftarrow \mathcal{Z}(\rho_{out}(\mathcal{S}_{\mathcal{A}}, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in}))\right] - \Pr\left[1 \leftarrow \mathcal{Z}(\rho_{out}(B, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in}))\right]\right| \leq \epsilon(\eta) \tag{3.5}$$

where ρ_{in} *is the joint input state of parties A and B, the auxiliary input state of the Adversary and Environment,* $\rho_{out}(\mathcal{S}_{\mathcal{A}}, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in})$ *corresponds to the final state of the Adversary when interacting with Simulator* $\mathcal{S}_{\mathcal{A}}$ *along with the output of honest party B in the ideal execution with Ideal Functionality* $\mathcal{F}$ *and* $\rho_{out}(B, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in})$ *corresponds to the final state of the Adversary when interacting with honest party B in the real protocol* Π *along with the output of the honest party B. The probability is taken over all executions of protocol* Π *and all possible input states produced by a* QPT*-limited Environment.*

The main advantage of using the framework of [66] is that it allows for a sequential composability property analogous to that of classical stand-alone models, which other *quantum* stand-lone models lack. This is captured by Theorem 3.4 from [66], recalled below. For further details on these definitions and properties, see [44, 66].

Theorem 3.1 (Sequential Composition Theorem)**.** *Let* Π *be a two-party protocol that calls another protocol* Γ *as a subroutine, with the restriction that, at any point, only one subroutine call to* Γ *be in*

progress and no other process is active during the call. Let Γ' *be a protocol that securely emulates* Γ *(in the sense of definition 3.7). The composed protocol, denoted* $\Pi^{\Gamma/\Gamma'}$*, is defined to be the protocol in which each invocation of* Γ *is replaced by an invocation of* Γ'*. Then* $\Pi^{\Gamma/\Gamma'}$ *securely emulates* Π*.*

Comments on the Model. Statistical security is defined similarly by quantifying over all quantum Adversaries and Environments (instead of QPT). Note that if the trace distance between output states (and auxiliary states) is bounded by ϵ, then the condition in Definition 3.7 is also verified for all Environments, which implies statistical security. We often use this characterisation instead. This can be formally expressed as:

$$\Delta(\rho_{out}(\mathcal{S}_{\mathcal{A}}, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in}), \rho_{out}(B, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in})) \leq \epsilon(\eta) \tag{3.6}$$

If $\epsilon(\eta) \sim C/\eta^k$ for some fixed exponent k and constant C, we say that the protocol is *inverse-polynomially-secure* against Malicious Adversaries, whereas if $\epsilon(\eta)$ is negligible in η we say that it is *fully-secure* against Malicious Adversaries. The main difference is that in the inverse-polynomial case the probability that Malicious Adversaries cheat successfully may be low but not negligible.

Stronger and more elegant definitions (Covert Adversaries) that might be adaptable to the quantum inverse-polynomial case can be found in [11] (in the classical case). This models real world situations where getting caught might have dire consequences for the parties, e.g. financial repercussions. By associating the appropriate cost to being caught, even if the probability of getting caught is close to 1 but not exponentially so, the deterrence might be still high enough to make cheating unappealing. For reasons specific to our constructions (namely the fact that measurement is irreversible and disturbs quantum states) they are not directly applicable here.

Any party may choose to abort by sending Abort at any moment, whether in the real protocol or ideal setting, which then forwards it to the other party and halts. Actions not explicitly described in an Ideal Functionality or Resource or sending invalid inputs results in them sending Abort to both players and then halting as well.

3.3.2 Abstract Cryptography Framework

Abstract Cryptography is a framework for defining and proving the security of cryptographic protocols, first introduced in [99, 98]. Its main advantage is that any system that follows the structure defined by the framework is inherently composable, in the sense that if two protocols are secure separately, the framework guarantees at an abstract level that their sequential or parallel composition is also secure. We refer the reader to [41] for a more in-depth presentation.

In this framework, the purpose of a secure protocol π is, given a number of available *resources* $\mathcal{R}$, to construct a new resource – written as $\pi\mathcal{R}$. This new resource can be itself reused in a future protocol. The actions of all honest players in a given protocol are represented as a sequence of efficient CPTP maps acting on their internal and communication registers. We use this framework in this thesis in the multi-party setting. An N-party quantum protocol is therefore described by $\pi = (\pi_1, \ldots, \pi_N)$ where π_i is the aforementioned sequence of efficient CPTP maps executed by party i (called the *converter* of party i). In the two-party setting these sets of CPTP maps correspond to the maps $\mathcal{E}_i$ and $\mathcal{F}_i$. A resource $\mathcal{R}$ is described as a sequence of CPTP maps with an internal state. It has input and output interfaces

describing which party may exchange states with it. An interface is said to be filtered if it is only accessible by a dishonest player. It works by having the party sending it a given state at one of its input interfaces, applying the specified CPTP map after all input interfaces have been initialised and then outputting the resulting state at its output interfaces in a specified order. Classical resources are modelled by considering that the input state is measured upon reception and the output is a computational basis state. These resources are the equivalent in this framework of the Ideal Functionalities of the Stand-Alone Model discussed above and we will use the two terms interchangeably. Useful resources/ideal functionalities are presented in the next section.

In order to define the security of a protocol, we need to give a pseudo-metric on the space of resources. We consider for that purpose a special type of converters called distinguishers which have as many input interfaces as the resources and which output a single bit. The distinguisher's aim is to discriminate between an execution with a resource $\mathcal{R}_1$ and another resource $\mathcal{R}_2$, each having the same number of input and output interfaces. It prepares the input, interacts with the resource according to its own possibly adaptive strategy, and decides whether it was interacting with one or the other. Two resources are said to be indistinguishable if no distinguisher can make this guess with good probability.

Definition 3.8 (Statistical Indistinguishability of Resources)**.** *Let $\epsilon(\eta)$ be a function of security parameter η and $\mathcal{R}_1$ and $\mathcal{R}_2$ be two resources with same input and output interfaces. The resources are ϵ-statistically-indistinguishable if, for all distinguishers $\mathcal{D}$, we have:*

$$\left|\Pr[b = 1 \mid b \leftarrow \mathcal{D}\mathcal{R}_1] - \Pr[b = 1 \mid b \leftarrow \mathcal{D}\mathcal{R}_2]\right| \leq \epsilon \tag{3.7}$$

We then write $\mathcal{R}_1 \underset{stat,\epsilon}{\approx} \mathcal{R}_2$.

The construction of a given resource $\mathcal{S}$ by the application of protocol π to resource $\mathcal{R}$ is captured by the fact that these resources are indistinguishable. More specifically, this captures the correctness of the protocol. The security is captured by the fact that the resources remain indistinguishable if we allow some parties to deviate in the sense that they are no longer forced to use the converters defined in the protocol but can use any other CPTP maps instead. This is done by removing the converters for those parties in Equation 3.7 while keeping only $\pi_H = \prod_{i \in H} \pi_i$ where H is the set of honest parties. The security is formalised as follows in Definition 3.9.

Definition 3.9 (Construction of Resources)**.** *Let $\epsilon(\eta)$ be a function of security parameter η. We say that an N-party protocol π ϵ-statistically-constructs resource $\mathcal{S}$ from resource $\mathcal{R}$ against adversarial patterns $\mathsf{P} \subseteq \wp([N])$ if:*

1. *It is correct: $\pi\mathcal{R} \underset{stat,\epsilon}{\approx} \mathcal{S}$*
2. *It is secure for all subsets of corrupted parties in the pattern $M \in \mathsf{P}$: there exists a simulator (converter) σ_M such that $\pi_{M^c}\mathcal{R} \underset{stat,\epsilon}{\approx} \mathcal{S}\sigma_M$*

We can now present the following General Composition Theorem (Theorem 1 from [99]).

Theorem 3.2 (General Composability of Resources)**.** *Let $\mathcal{R}$, $\mathcal{S}$ and $\mathcal{T}$ be resources, α, β and id protocols (where protocol id does not modify the resource it is applied to). Let $\circ$ and $|$ denote respectively the sequential and parallel composition of protocols and resources. Then the following implications hold:*

- *The protocols are* sequentially composable*: if* $\alpha\mathcal{R} \underset{stat,\epsilon_\alpha}{\approx} \mathcal{S}$ *and* $\beta\mathcal{S} \underset{stat,\epsilon_\beta}{\approx} \mathcal{T}$ *then* $(\beta \circ \alpha)\mathcal{R} \underset{stat,\epsilon_\alpha+\epsilon_\beta}{\approx} \mathcal{T}$
- *The protocols are* context-insensitive*: if* $\alpha\mathcal{R} \underset{stat,\epsilon_\alpha}{\approx} \mathcal{S}$ *then* $(\alpha \mid \mathsf{id})(\mathcal{R} \mid \mathcal{T}) \underset{stat,\epsilon_\alpha}{\approx} (\mathcal{S} \mid \mathcal{T})$

Combining the two properties presented above yields concurrent composability (the distinguishing advantage cumulates additively as well).

The computational versions of these definitions and theorem are obtained by quantifying over QPT parties. Composing a statistically-secure protocol with a computationally-secure protocol is possible provided that the simulator for the statistically-secure one runs in expected polynomial-time. The resulting protocol is of course only computationally-secure.

3.3.3 Ideal Functionalities and Resources

In this section, we present the ideal functionalities that appear throughout the thesis and which have been previously defined in the literature. These correspond to the primitives presented informally in Section 3.1.1. New resources and ideal functionalities introduced in this thesis for the first time are presented in their specific chapters. The term resource or ideal functionality is employed according to the framework of security that they are used in later.

We suppose that the following rules apply to all primitives formalised below. Honest parties send their prescribed input to the Ideal Functionality, corrupted parties send any input given to them by the Adversary, computed (efficiently in the computational setting) from the prescribed input and the Adversary's internal state. All resources allow malicious players to force a unanimous abort of honest players by simply not sending their inputs (there is no guaranteed output delivery). However, no functionality allows malicious parties to selectively choose which honest party will abort or receive their outputs: either all honest parties terminate successfully or none do. The exact value of the leakage l_ρ, alluded to by some functionalities, is a publicly known function of inputs and computation which is specified by each protocol. In this thesis, this leakage is simply an upper-bound on the size of the quantum circuit implementing the unitary and length of the Parties' input.

3.3.3.1 Building Blocks

We start by presenting simple Ideal Functionalities and Resources (1 to 11) that are later used as building blocks for constructing larger and more complex ones.

Resource 1 Authenticated Classical Channel

Inputs: The Sender inputs a message m. The Receiver has no input. The Eavesdropper inputs two bits $(e, c) \in \{0,1\}^2$. This last interface is filtered and set to $(0, 0)$ in the honest case.
Computation by the Resource: If $e = 1$, it sends m to the Eavesdropper. If it receives $c = 1$ from the Eavesdropper, it sends Abort to the Receiver. Otherwise it sends m to the Receiver.

3.3.3.2 Target Resources

These are the functionalities that our protocols will construct (12 to 14).

Resource 14 is identical to the one from [40] apart from the explicit introduction of an Eavesdropper, i.e. a party with no input and no output. This allows for an easier account and analysis of the information

Ideal Functionality 2 Confidential Classical Channel

Inputs: The Sender inputs a message m. The Receiver has no input. The Eavesdropper has an auxiliary input $\hat{m}$.
Computation by the Functionality: It sends the bit-length $\#m$ of message m to the Eavesdropper. If it has not received anything from the Eavesdropper, it sends m to the Receiver. Otherwise if it has received message $\hat{m}$ from the Eavesdropper over $\#m$ bits, it then sends it to the Receiver.

Resource 3 Secure Classical Channel

Inputs: The Sender inputs a message m. The Receiver has no input. The Eavesdropper inputs two bits $(e, c) \in \{0,1\}^2$.This last interface is filtered and set to $(0, 0)$ in the honest case.
Computation by the Resource: If $e = 1$, it sends $\#m$ to the Eavesdropper. If it receives $c = 1$ from the Eavesdropper, it sends Abort to the Receiver. Otherwise it sends m to the Receiver.

Resource 4 Insecure Quantum Channel

Inputs: The Sender inputs a quantum state ρ. The Receiver has no input. The Eavesdropper inputs a bit $c \in \{0,1\}$ and quantum state $\tilde{\rho}$. This last interface is filtered and c is set to 0 in the honest case.
Computation by the Resource: If $c = 1$, it sends ρ to the Eavesdropper. In that case it waits for a state $\tilde{\rho}$ and forwards it to the Receiver. Otherwise it sends ρ to the Receiver.

Resource 5 Confidential Quantum Channel

Inputs: The Sender inputs a quantum state ρ. The Receiver has no input. The Eavesdropper inputs a bit $c \in \{0,1\}$ and quantum state $\tilde{\rho}$. This last interface is filtered and c is set to 0 in the honest case.
Computation by the Resource: If $c = 1$, it sends $\#\rho$ to the Eavesdropper. In that case it waits for a state $\tilde{\rho}$ and forwards it to the Receiver. Otherwise it sends ρ to the Receiver.

Resource 6 Common Reference String

- **Public Information:** The BPP algorithm Setup that is used to produce the reference string.
- **Inputs:** Both players input the dummy input λ.
- **Computation by the Resource:** The Resource generates $\mathscr{C} \leftarrow \mathsf{Setup}$ and sends it to both players.

Ideal Functionality 7 Key Distribution

Inputs: Parties P_1 and P_2 have as input the size n of the key. The Eavesdropper has no input.
Computation by the trusted party: It samples uniformly at random $k \in_R \{0,1\}^n$ and sends k to P_1 and P_2. It sends n to the Eavesdropper.

Resource 8 Two-Party Coin-Tossing

Public Information: Distribution $\mathcal{D}$ from which the value is sampled. Party $j \in \{1, 2\}$ receives the output first.
Inputs: Parties P_1 and P_2 each have as input a bit a_i. P_1 has an additional input c. These interfaces are filtered and the bits set to 0 in the honest case.
Computation by the Resource:

- If the resource receives at least one $a_i = 1$ on either interface, it outputs Abort to both interfaces.
- Otherwise:
  1. The resource samples $S \leftarrow \mathcal{D}$ and sends it to P_1.
  2. If it receives $c = 1$ from P_1, it outputs Abort to P_2. Otherwise it sends S to P_2.

Ideal Functionality 9 1-out-of-2 String Oblivious Transfer

Inputs: P_1 has as input $(x_0, x_1) \in \{0,1\}^{2n}$ for known n and P_2 has as input $b \in \{0,1\}$.
Computation by the Functionality: The Ideal Functionality sends x_b to P_2.

Resource 10 Classical Secure Multi-Party Computation

Inputs: The N parties each input a value x_j and a common classical computation C. They each have a filtered input bit o_j (set to 0 in the honest case) indicating whether they receive their inputs in advance.
Computation by the Resource: The Resource computes $y = C(x_1, \ldots, x_N)$. It first sends y to all parties j such that $o_j = 1$. Then it sends y to all other parties.

Resource 11 Blind Delegated Quantum Computation, Taken from [41]

Inputs:

- The Client inputs a quantum state ρ_C and the classical description of a unitary U.
- The Server chooses whether or not to deviate. This interface is filtered by two control bits (e, c) (set to 0 by default for honest behaviour). If deviating it also inputs a state ρ_S and CPTP map $\mathcal{E}$.

Computation by the Resource:

- If $e = 1$, the Resource sends the leakage l_ρ to the Server's interface and awaits further input from the Server; if it receives $c = 1$ and $(\rho_S, \mathcal{E})$, the Resource outputs $\mathcal{E}(\rho_{CS})$ at the Client's output interface, where ρ_{CS} is a joint state between Client and Server (these may be entangled).
- Otherwise it outputs $U(\rho_C)$ at the Client's output interface.

Resource 12 Verifiable Blind Delegated Quantum Computation, Taken from [41]

Inputs:

- The Client inputs a quantum state ρ_C and the classical description of a unitary U.
- The Server chooses whether or not to deviate. This interface is filtered by two control bits (e, c) (set to 0 by default for honest behaviour).

Computation by the Resource:

- If $e = 1$, the Resource sends the leakage l_ρ to the Server's interface and awaits further input from the Server; if it receives $c = 1$, the Resource outputs $|\mathsf{Abort}\rangle\langle\mathsf{Abort}|$ at the Client's output interface.
- Otherwise it outputs $U(\rho_C)$ at the Client's output interface.

Ideal Functionality 13 Classical Input Two-Party Quantum Computation

Inputs:

- P_1 and P_2 have classical inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ respectively.
- They have agreed on a common description of unitary U applied on inputs of length n along with k ancillae, as well as the position of output qubits for each player (as registers $\mathcal{O}_1$ and $\mathcal{O}_2$).

Computation by the Functionality: The Ideal Functionality initialises its internal registers $(\mathcal{X}, \mathcal{Y}, \mathcal{W})$ to $(|x\rangle, |y\rangle, |0\rangle^{\otimes k})$ and applies U to these registers. It sends output register $\mathcal{O}_1$ to P_1 first and then $\mathcal{O}_2$ to P_2.

Resource 14 Secure Multi-Party Quantum Computation

Inputs:

- N players send each a quantum register $\mathcal{X}_j$ which contains their respective part of a collectively possessed ρ_{inp}, and the classical description of a joint unitary U they wish to apply to ρ_{inp}. They can each input two bits o_j and c_j as a filtered interface.
- The Eavesdropper can input two bits e and c as a filtered interface, both set to 0 in the honest case.

Computation by the Resource:

- If $e = 1$, the Resource sends the leakage l_ρ to the Eavesdropper's interface.
- If $c = 1$ or there exists j such that $c_j = 1$, the Resource sends Abort to all players.
- Otherwise it computes $U(\rho_{inp})$. If there exists $j \in [N]$ such that $o_j = 1$, it sends qubit j to the interface of player j. It then sends the output qubits to all other players in a similar fashion.

the Server might obtain when the computation is delegated since, despite being active in the protocol, the Server has no legitimate input nor output.

3.3.4 Quantum One-Time Pad Security

We present here the Quantum One-Time Pad Protocol based on the Quantum One-Time Pad encryption, which aims to construct a Confidential Quantum Channel (Ideal Resource 5) from an Insecure Quantum Channel (Ideal Resource 4) and a Secure Classical Channel (Ideal Resource 3). We usually abstract the channels in later protocols but write them down explicitly for this simple setting.

Protocol 3 Quantum One-Time Pad

Inputs: The Sender has as input a quantum register containing n qubits. The Receiver and the Eavesdropper have no input.

Protocol:

1. The Sender samples uniformly at random a key $k = (k_Z, k_X) \in_R \{0,1\}^{2n}$ with $\#k_Z = \#k_X = n$ and applies $\mathsf{Z}^{k_Z}\mathsf{X}^{k_X}$ to its quantum register (where the operation $\mathsf{Z}^{k_Z^i}\mathsf{X}^{k_X^i}$ is applied to qubit i for single-qubit Paulis $\mathsf{Z}^{k_Z^i}$ and $\mathsf{X}^{k_X^i}$ controlled by the key bits k_Z^i and k_X^i respectively).
2. It sends the quantum register to the Eavesdropper through an Insecure Quantum Channel Ideal Resource and it sends (k_X, k_Z) to the Receiver through a Secure Classical Channel.
3. The Eavesdropper transmits the received quantum state to the Receiver through another Insecure Quantum Channel Ideal Resource.
4. The Receiver applies $\mathsf{X}^{k_X}\mathsf{Z}^{k_Z}$ to the received quantum register and keeps the resulting state as its output.

The following statement captures the security of Protocol 3.

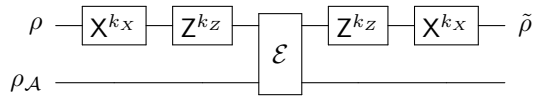
Theorem 3.3 (Security of Quantum One-Time Pad)**.** *Protocol 3 perfectly constructs Ideal Resource 5 (statistical security with distinguishing advantage* 0*).*

Proof. The correctness of the protocol trivially stems from the properties of the Pauli operations (namely $\sigma^2 = \mathsf{I}$ for any Pauli σ). The only security that needs to be proven is against a malicious Eavesdropper.

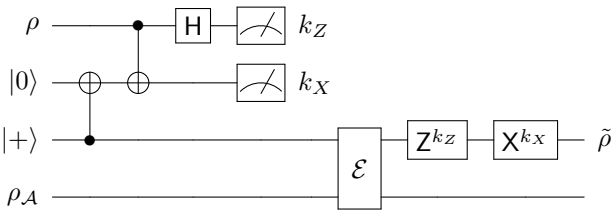
The Eavesdropper receives in the real protocol a quantum state from one Insecure Quantum Channel and sends one quantum state of the same size to another. In all generality, it applies in between these

transmissions an arbitrary CPTP map on the received state and its internal state. In the ideal world, the Simulator receives the size of the state from the Confidential Quantum Channel Ideal Resource and must replicate the view of the Adversary and honest players by: sending a state to the Eavesdropper, receive another state in return and send a state to the Confidential Quantum Channel Ideal Resource.

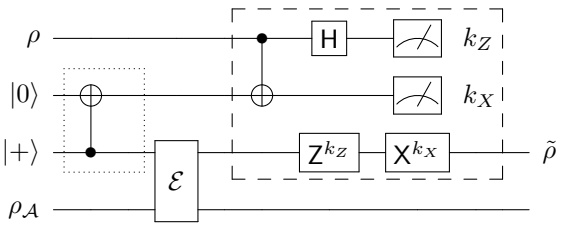
We start by representing the real protocol as a quantum circuit for fixed keys (k_Z, k_X) and a single qubit, where $\mathcal{E}$ is a general CPTP map and $\rho_{\mathcal{A}}$ is the Adversary's internal state:



In the circuit above, the first two operations are performed by the Sender, $\mathcal{E}$ is applied by the Eavesdropper and the final two Paulis correspond to the Receiver's decryption. Note that it is possible, although slightly convoluted, for the Sender to perform the encryption via a quantum teleportation: it creates first an EPR-pair, applies a CNOT with its input as control and one qubit of the EPR-pair as the target followed by a Hadamard on its input and measures these two qubits in the computational basis. The Sender defines the measurement results to be (k_Z, k_X), in which case, as explained in Section 2.2.3.2, the state of the unmeasured qubit of the EPR-pair is $\mathsf{Z}^{k_Z}\mathsf{X}^{k_X}(\rho)$ for input state ρ. This perfectly equivalent to the initial protocol and the resulting circuit is represented as follows:



In the circuit above, the first CNOT creates the EPR-pair, the second CNOT and H along with the measurement in the computational basis perform the teleportation. We then remark that it is equivalent from the point of view of all parties if the measurements are postponed until after the Receiver has received the state from the Eavesdropper (otherwise this would violate the No-Communication result from Section 2.2.3.4). The Sender can simply send the third qubit to the Eavesdropper, wait for confirmation that the Receiver has obtained the state, measure its remaining qubits and transfer the result via the Secure Classical Channel:



Finally we can isolate from the circuit above the operations of the Simulator – dotted box – and the Ideal Resource – dashed box – for a single qubit (if multiple qubits are sent these operations are applied for all qubits in parallel). The Simulator prepares an EPR-pair and sends half to the Eavesdropper. After receiving the reply of the Eavesdropper, it sends the other half of the EPR-pair and the third qubit to the Ideal Resource. In the mean-time, the Sender sends its input qubit to the Ideal Resource.

The Resource applies the CNOT and H and measures the first two qubits in the computational basis, obtaining (k_Z, k_X). It applies $\mathsf{X}^{k_X}\mathsf{Z}^{k_Z}$ to the third qubit and sends this qubit to the Receiver as its output.

The indistinguishability comes from the fact that the input/output relation is preserved via the transformations above and the fact that the Eavesdropper cannot distinguish between half an EPR-pair and a qubit encrypted via a Quantum One-Time Pad since both are perfectly mixed states from its point of view (Equation 2.22).

■

This proof is important as its principle is the basis for the proof of security of more complicated protocols presented later in the thesis. In particular, using EPR-pairs will be again crucial to transferring the deviation of a party from a state that is independent of the secret parameters and messages to another which may depend on it.

3.4 Local Criteria of Security for Delegated Quantum Computation

We present here a few definitions and results for the security of Delegated Quantum Computing (DQC), a functionality where a Client with limited quantum capabilities (or non in some cases) wishes to delegate a quantum computation to a powerful but distrustful quantum Server. The Client wants its computation and input to remain hidden from the Server but at the same time be able to verify that the Server has performed the computation as instructed.

The following security criteria are said to be local since they only depend on the protocol and do not consider an outside Environment. They include local-correctness (if the Server is honest, the protocol terminates by outputting the correct outcome), local-blindness (which essentially says that the state of the Server at the end of the protocol is close to a state where the Server has applied an operation to its own internal state), and input-independent local-verifiability (the Client can verify with its own internal state that the computation was carried out correctly and leaking the information of whether the Client has aborted or not does not give more information to the Server about the Client's input or computation).

We start by defining the correctness of a protocol:

Definition 3.10 (ϵ_{cor}-local-correctness)**.** *Let $\mathcal{P}_{AB}$ be the protocol as defined above (with the honest CPTP maps for players A and B). We say that such a protocol implementing $\mathcal{U}$ is ϵ_{cor}-locally-correct if for all input states ρ we have:*

$$\Delta(\mathrm{Tr}_B \circ \mathcal{P}_{AB}(\rho), \mathcal{U}(\rho)) \leq \epsilon_{cor} \tag{3.8}$$

Next we define local-blindness, which captures the fact that the state at the end of the protocol is indistinguishable for the Adversary from one which it could have generated on its own without having access to the honest player's state.

Definition 3.11 (ϵ_{bl}-local-blindness)**.** *We say that a protocol $\mathcal{P}_{AB}$ as defined above is ϵ_{bl}-locally-blind if there exists a CPTP map $\mathcal{F}' : L(\mathcal{B}) \to L(\mathcal{B})$ such that, for all input states $\rho \in L(\mathcal{A} \otimes \mathcal{B})$, we have:*

$$\Delta(\mathrm{Tr}_A \circ \mathcal{P}_{AB}(\rho), \mathcal{F}' \circ \mathrm{Tr}_A(\rho)) \leq \epsilon_{bl} \tag{3.9}$$

On the other hand, local-verifiability is satisfied if any deviation by the Adversary which leads to a wrong output will cause the honest party to abort with overwhelming probability.

Definition 3.12 (ϵ_{ver}-local-verifiability)**.** *Let ρ_{in} be the input state of A in the protocol implementing unitary U. For all CPTP maps $\{\tilde{\mathcal{F}}_j\}$ (corresponding to the deviation of player B^*), let $\tilde{\rho}_{out}\Big(\{\tilde{\mathcal{F}}_j\}, \rho_{in}\Big)$ be the output of honest party A in the protocol. For a given $p_{ok} \in [0, 1]$, the ideal output is defined as:*

$$\rho_{ideal}(\rho_{in}) := p_{ok} U(\rho_{in}) + (1 - p_{ok}) \, |\mathsf{Abort}\rangle\langle\mathsf{Abort}| \tag{3.10}$$

The protocol is ϵ_{ver}-locally-verifiable for A if:

$$\Delta\Big(\tilde{\rho}_{out}\Big(\{\tilde{\mathcal{F}}_j\}, \rho_{in}\Big), \rho_{ideal}(\rho_{in})\Big) \leq \epsilon_{ver} \tag{3.11}$$

In the case where the corrupted party has an input and output to the computation, it can always modify its input and perform computations on its final state, meaning that the definition should account for those possibilities with a modified ideal state. Let $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ be respectively deviation CPTP-maps acting on the input and output systems of Malicious B^* an some auxiliary register. Now the input $\rho_{in,aux}$ is a joint input potentially entangled with the auxiliary state of B^* and for a given $p_{ok} \in [0, 1]$, the ideal output is defined as:

$$\rho_{ideal}(\rho_{in}) := p_{ok}(\mathsf{I}_A \otimes \mathcal{D}_{out}) \cdot U \cdot (\mathsf{I}_A \otimes \mathcal{D}_{in}) \cdot (\rho_{in,aux}) + (1 - p_{ok}) \, |\mathsf{Abort}\rangle\langle\mathsf{Abort}| \tag{3.12}$$

Furthermore, we say that the verification procedure is independent of the inputs of player A if player B could have deduced from its own internal registers whether the protocol will lead to an abort for player A or not. This takes care of the fact that some protocols might fail to specify in their description the fact that a dishonest player will know at the end that the protocol has been aborted by the honest player and omit it later in their stand-alone security analysis (this does not lead to a decrease in security in the stand-alone case, as it may be that player B simply does not know at the end whether its attacks succeeded or not). This is captured by the following definition:

Definition 3.13 (ϵ_{ind}-independent verification)**.** *Let $\mathcal{P}_{AB}$ be the protocol as defined above and let $\mathcal{Q}_{A\bar{B}} : L(\mathcal{A} \otimes \bar{\mathcal{B}}) \to L(\mathcal{A} \otimes \bar{\mathcal{B}})$ be a CPTP map which, conditioned on $\mathcal{A}$ containing the state $|\mathsf{Abort}\rangle$, switches the state in $\bar{\mathcal{B}}$ from $|\mathsf{Ok}\rangle$ to $|\mathsf{Abort}\rangle$ (and does nothing in the other cases). The register in $\bar{\mathcal{B}}$ starts in state $|\mathsf{Ok}\rangle$. We say that such a protocol's verification procedure is ϵ_{ind}-independent from player A's input if there exists CPTP maps $\mathcal{F}'_i : L(\mathcal{C} \otimes \mathcal{B} \otimes \bar{\mathcal{B}}) \to L(\mathcal{C} \otimes \mathcal{B} \otimes \bar{\mathcal{B}})$ such that:*

$$\Delta(\mathrm{Tr}_A \circ \mathcal{Q}_{A\bar{B}} \circ \mathcal{P}_{AB}(\rho), \mathrm{Tr}_A \circ P'_{AB\bar{B}}(\rho)) \leq \epsilon_{ind} \tag{3.13}$$

where (with $\mathcal{E}_i$ being the CPTP map of honest player A for round i):

$$P'_{AB\bar{B}} := \mathcal{E}_1 \circ \mathcal{F}'_1 \circ \ldots \circ \mathcal{E}_n \circ \mathcal{F}'_n \tag{3.14}$$

Finally, it is proven in [41] that the security of a Verifiable Blind Delegated Quantum Computation Protocol can be reduced to these local-criteria usually used in stand-alone proofs. We now state the Local-Reduction Theorem from [41]:

Theorem 3.4 (Security Reduction to Local Criteria)**.** *If a protocol implementing a unitary transformation is ϵ_{cor}-locally-correct, ϵ_{bl}-locally-blind and ϵ_{ver}-locally-verifiable with ϵ_{ind}-independent verification for all inputs $\psi_{A_C A_Q}$, where register $\mathcal{A}_C$ is classical while $\mathcal{A}_Q$ is quantum, then it constructs $\mathcal{S}^{verif}_{blind}$ within $\epsilon = max\{\delta 2^{2N}, \epsilon_{cor}\}$, where $N := \#\mathcal{A}_Q$ is the number of qubits contained in the Client's quantum input register and:*

$$\delta := 4\sqrt{2\epsilon_{ver}} + 2\epsilon_{bl} + 2\epsilon_{ind} \tag{3.15}$$

Essentially, while a simple sequentially-composable verifiable blind computation protocol is ϵ_{bl}-locally-blind and ϵ_{ver}-locally-verifiable, the price to pay for composability is $\epsilon = max\{\delta N^2, \epsilon_{cor}\}$, along with the additional independence property. On one hand, it means that, if a protocol has been shown to have these properties separately, its security bound will be higher in a non-composable setting (compared to using this reduction). On the other hand, it may be more efficient to create a protocol which directly satisfies the security properties of the AC Framework (as the reduction in the case of quantum inputs is not tight).

3.5 Core Cryptographic Protocols

This Section presents more complex protocols that will be used as stepping stones for our later constructions, namely Delegated Quantum Computations (both Blind and Verifiable variants), Yao's classical (and classic) Protocol from [135] and Universal Thresholdiser, whose purpose it to emulate a Classical SMPC.

3.5.1 Hiding Delegated Quantum Computations in MBQC

An MBQC computation between Client and Server as described in Section 2.3 can be totally hidden from the Server by using the following observation: if, instead of the Server preparing each qubit in the graph in the state $|+\rangle$, the Client sends $|+_{\theta(v)}\rangle$ with $\theta(v) \in_R \Theta$, then measuring the qubits in a similarly rotated basis, obtained by adding $\theta(v)$ to the measurement angle, has the same result as the initial computation. If the Client keeps the angle $\theta(v)$ hidden from the Server, the Server is completely blind on what computation is being performed. The angle $\theta(v)$ acts as a One-Time Pad for the measurement angle $\phi'(v)$. Nevertheless, because the Server could always measure the received qubits, it would still learn 1 bit of information about the angle $\theta(v)$, which can take the 8 values from set Θ and so consists of 3 bits. To prevent this, another parameter $r(v)$ is added for each qubit. The parameter $r(v)$ serves as a One-Time-Pad for the measurement outcome: if $b(v)$ is now the outcome returned by the Server, then we have $s(v) = b(v) \oplus r(v)$ with $s(v)$ defined as above.

If $v \in I$ is a quantum input, it is protected by a Quantum One-Time Pad operation $\mathsf{Z}(\theta(v))\mathsf{X}^{a(v)}$ for $a(v) \in_R \{0,1\}$ applied by the Client before being sent to the Server.[6] The measurement angles are further adapted to account for $a(v)$: $\hat{s}^X(v) = a(v) \oplus s^X(v)$ and $\hat{s}^Z(v) = a(f^{-1}(v)) \oplus s^Z(v)$ (where f is the flow of the computation, and a has been extended so that $a(v) = 0$ for $v \in I^c$ and $a(f^{-1}(v)) = 0$ for vertices $v \notin \text{range}(f)$).

The resulting measurement angle sent to the Server with all parameters taken into account is then (for $\phi'(v)$ defined using the new $\hat{s}^X(v)$ and $\hat{s}^Z(v)$):

$$\delta(v) = \phi'(v) + \theta(v) + r(v)\pi \tag{3.16}$$

In short, the Client sends randomly rotated qubits that appear maximally mixed to the Server and that become the resource state once entangled. It then guides the computation with a set of classical instructions that are adapted to the effectively prepared resource state but still look perfectly random to the Server. It is the combination of these two parts (quantum state preparation and classical instructions) that leads to the desired blind computation. This idea was first formalized in the *Universal Blind Quantum Computation* (UBQC) Protocol in [21].

For completeness we give the protocol for UBQC (Protocol 4) from [21]. We assume that a standard labelling and measurement ordering of the vertices of the graph is known to both the Client and the Server. Its security properties are given in the next section.

Protocol 4 Universal Blind Quantum Computation - Taken from [21]

Public Information: Description of a graph $G = (V, E, I, O)$ with input vertices I and output vertices O and an ordering over vertices.
Inputs: The Client inputs a quantum state ρ_{inp} and the classical description of a unitary U as a flow $(f, \preceq)$ on G and default measurement angles $\{\phi(v)\}_{v \in O^c}$. The Server has no input.
Protocol:

1. For each input qubit $v \in I$, the Client samples a Q-OTP key $\theta(v) \in_R \Theta$ and $a(v) \in_R \{0,1\}$ and encrypts qubit v using $\mathsf{Z}(\theta(v))\mathsf{X}^{a(v)}$. The Client prepares each qubit $v \in I^c$ in state $|+_{\theta(v)}\rangle$, with random $\theta(v) \in_R \Theta$. It then sends these qubits to the Server and initialises a string $\mathbf{s}$ to 0 representing the measurement outcomes on G.
2. The Server receives the qubits one-by-one and applies the entangling operations CZ that correspond to the edges of the graph G (if two two vertices are joined by an edge in the graph, a CZ operation is applied to the qubits corresponding to these vertices).
3. For each qubit $v \in O^c$, following the partial order of the flow:
   a) The Client generates $r(v) \in_R \{0,1\}$, calculates $\delta(v)$ according to Equation 3.16 and sends it to the Server.
   b) The Server measures in the $\{|+_{\delta(v)}\rangle, |-_{\delta(v)}\rangle\}$ basis and returns to the Client outcome $b(v)$.
   c) The Client sets the value of $s(v)$ in $\mathbf{s}$ to $b(v) + r(v)$.
4. The Server returns all qubits that correspond to vertices $i \in O$ to the Client on which the Client performs final Pauli correction: $\mathsf{Z}^{\hat{s}^Z(v)}\mathsf{X}^{\hat{s}^X(v)}$. It sets these qubits as its output.

We recall here the following security statement of the UBQC Protocol from [21].

[6] The initial presentation of this idea [21] applies the Q-OTP in the reversed order $\mathsf{X}^{a(v)}\mathsf{Z}(\theta(v))$, which is a typo since the operations would need to be commuted so that $\mathsf{Z}(\theta(v))$ can be cancelled by the $\theta(v)$ contained in the $\delta(v)$. Without this correction, the commutation of $\mathsf{X}^{a(v)}$ and $\mathsf{Z}(\theta(v))$ would turn $\theta(v)$ into $(-1)^{a(v)}\theta(v)$, which would not cancel out with the one from the measurement.

Theorem 3.5 (Local-Security of UBQC, Taken from [21]). *The UBQC Protocol 4 is perfectly locally-correct (Definition 3.10 for $\epsilon_{cor} = 0$) and perfectly locally-blind (Definition 3.11 for $\epsilon_{bl} = 0$).*

They prove furthermore in [41] that the UBQC Protocol 4 is perfectly composably-secure.

Theorem 3.6 (Composable Security of UBQC, Taken from [41]). *The UBQC Protocol 4 perfectly constructs (Definition 3.9 for statistical security with $\epsilon = 0$) the Blind Delegated Quantum Computation Ideal Resource 11 from Insecure Quantum Channels (Ideal Resource 4) and Authenticated Classical Channels (Ideal Resource 1).*

3.5.2 Verifying MBQC Through Trap Insertion

In UBQC, the Server is not forced to follow the instructions and the Client can not verify if the computation is done correctly, but a modified version of the protocol allows for such verification. The central idea is to include *trap qubits* at positions unknown to the Server [53]. These qubits should have deterministic outcome if measured in the correct basis, remain undetectable by the Server, and not affect the computation.

To do this, the Client can send *dummy qubits*, meaning qubits randomly initialized in states $\{|0\rangle, |1\rangle\}$ instead of the usual $|+_\theta\rangle$. This has the effect of breaking the graph at this vertex, removing it from G along with any attached edges. Sending such dummies for all neighbours of a vertex isolates it from the rest of the graph, creating a trap. If measured in the same basis as the one used for their preparation, these traps yield deterministic outcomes while being undetectable by the Server. The latter is due to the fact that dummies appear as maximally mixed qubits from the Server's perspective and are thus no different than regular randomly chosen $|+_\theta\rangle$ states.

This idea was introduced in [53] and later optimised by different protocols, such as [72, 78]. The structure of the protocol from [78] makes it more adapted for 2PQC: the construction is local and therefore the Server knows which qubits corresponds to its input; which is why we use it here as the basis of our protocol. We became aware recently of a new protocol [134] achieving the same properties as [78] with better overhead and believe it could be combined with the same techniques as the ones presented in this thesis.

We describe here the trap insertion technique of [78], which relies on the Dotted-Triple Graph construction. This construction amounts to enlarging the initial graph so that it is possible to randomly choose one of the sub-graphs corresponding to the to-be-performed computation, isolate it from the rest of the graph using dummy qubits in the $|0\rangle$ or $|1\rangle$ state, and place trap qubits on remaining positions. We start by giving the definition of a Dotted-Triple Graph of base-graph used in the UBQC computation without traps.

Definition 3.14 (Dotted-Triple Graph, Taken from [78]). *Let $G = (V, E)$ be a graph with vertex set V and edge set E, the corresponding Dotted Triple-Graph $DT(G)$ is constructed in the following way:*

1. *For each vertex $v \in V$, we define a set of three new vertices $P(v) = \{p^1(v), p^3(v), p^3(v)\}$, called* primary *vertices.*
2. *For each edge $(v, w) \in E$ of the base-graph that connects vertices v and w in G, we introduce a set of nine edges $E_{(v,w)}$ so that all three vertices in the set $P(v)$ are connected to all vertices in the set $P(w)$.*

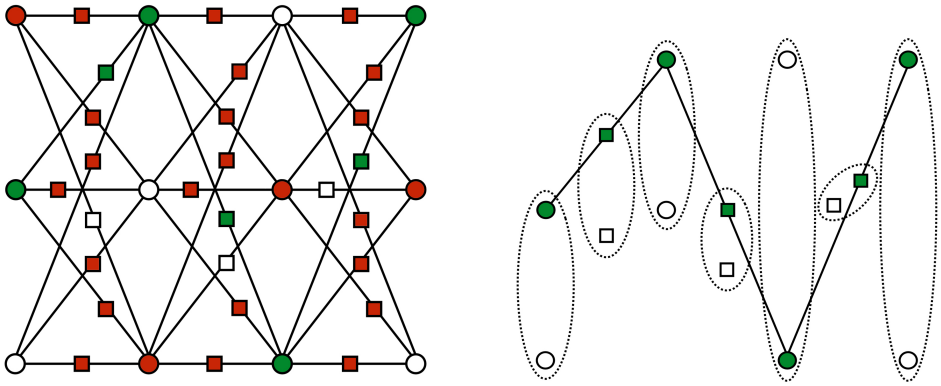
3. *We replace every edge in the resulting graph with a new vertex connected to the two vertices originally joined by that edge. These new vertices are called* added *vertices.*

The edge or vertex of the initial graph G that each vertex v of $DT(G)$ originates from is called its base-location.

By inserting dummy qubits among the added qubits we can break any DTG in three copies of the same base-graph: one is used for the computation while the other two are traps. Furthermore for each vertex base-location the choice of where to break the graph is independent from other vertex base-locations and can be made in advance by the Client. The Server remains totally ignorant of this choice. This choice is called *trap-colouring* (Definition 3.15, see Figure 3.1 for an example of such a colouring).

Definition 3.15 (Trap-Colouring, Adapted from [78])**.** *We define trap-colouring to be an assignment of one colour (green, white or red) to each of the vertices of the Dotted Triple-Graph that is consistent with the following conditions:*

1. *In each primary set $P(v)$ there is exactly one vertex coloured of each colour.*
2. *The colouring of added vertices is fixed by the choice above in the following way: green primary vertices are joined by a green added vertex, red primary vertices are joined by a white added vertex and the other added vertices are red.*



(a) Trap-colouring chosen by the Client. Green: computation qubits; White: trap qubits; Red: dummy qubits.

(b) After entangling, the breaking operation defined by the dummy qubits will reduce the graph in (a) to the computation graph and for each vertex a corresponding trap qubits.

Figure 3.1: Dotted-triple-graph for one-dimensional base graph of four qubits. Circles: primary vertices (base-location : vertex of the base-graph); Squares: added vertices (base-location : edge of the base-graph). Adapted from [78].

By setting $\phi(v) = 0$ for added qubits v, the flow, Past of qubit v and Influence-past of qubit v can all be extended to the Dotted-Triple-Graph construction, with the result that each qubit depends only on a constant number of previous measurements (see [78]). The protocol then implements the original MBQC computation on the sub-graph of $DT(G)$ that is used for computation (represented with

green vertices), while the Server learns nothing about the input/output nor computation. Moreover, the random placement of the traps guarantees that a deviating Server will be detected with non-zero probability by the Client from the corruption of trap measurement outcomes. Exponential amplification of this probability can be achieved by incorporating fault-tolerant techniques as described in [53].

Given the Dotted-Triple-Graph corresponding to the base-graph G, the Verifiable Blind Quantum Computation (VBQC) Protocol can be summarised as follows:

1. The Client chooses the trap-colouring of the Dotted-Triple Graph, keys for its inputs and secret parameters for the rest of the qubits in the DTG.
2. It encrypts its inputs as in UBQC, prepares the resource single qubit states corresponding to the description chosen above and sends them along with its inputs to the Server (in order). These states are (with D the set of dummy qubits in $DT(G)$):
   - For all $v \in D$: $|d(v)\rangle$, with $d(v) \in_R \{0,1\}$.
   - For all $v \notin D$: $\prod_{\tilde{v} \in N_G(v) \cap D} \mathsf{Z}^{d(\tilde{v})} \left|+_{\theta(v)}\right\rangle$, with $\theta(\tilde{v}) \in_R \Theta$.
3. The Server entangles them according to the publicly known structure of the graph (one CZ per edge in $DT(G)$).
4. The Client adaptively instructs the Server to measure all non-output qubits as in UBQC, and the Server returns the measurement outcomes. The measurement angle of dummies is chosen uniformly at random.
5. The Server returns to the Client all qubits that correspond to vertices at output base-locations to the Client. The Client measures the output trap qubits t with angle $\delta(t) = \theta(t) + r(t)\pi$.
6. The Client checks that the traps have been measured correctly (if $b(t) = r(t)$) and aborts if any failed. It computes encryption keys corresponding to Quantum One-Time-Pad of the output as in UBQC.
7. It decrypts the Quantum One-Time Pad and accepts the final quantum states as its output.

In the Dotted-Triple Graph used in the VBQC Protocol, the past of each qubit is not explicitly known to the Server because it does not know the positions of the traps and dummies. We therefore define:

Definition 3.16 (Extended Past and Influence-Past)**.** *We define the* Extended-Past *$EPast(v)$ of qubit v to be the set of all qubits that for some trap-colouring are in the past of v. We define the* Extended-Influence-Past *of qubit v via a similar extension.*

This new set is also finite and bounded by a constant because only qubits in neighbouring base-locations and their own neighbours can influence the δ of qubit v.

For completeness we give the Verifiable Blind Quatum Computation Protocol with Dotted-Triple Graph from [78]. As previously, we assume that a standard labelling and measurement ordering of the vertices of the dotted triple-graph $DT(G)$ is known to both the Client and the Server.

The VBQC Protocol inherits directly the correctness and blindness of the UBQC Protocol. We also restate here the local-verifiability of the VBQC Protocol as Theorem 3.7.

Theorem 3.7 (Local-Verifiability of VBQC, Taken from [78])**.** *Assume that the MBQC computation is encoded using a fault-tolerant encoding that corrects or detects δ errors when the computation is*

Protocol 5 VBQC using $DT(G)$ (with Fault-Tolerant Encoding) - Taken from [78]

Public Information: Description of a graph $G = (V, E, I, O)$ with input vertices I and output vertices O and an ordering over vertices, and corresponding dotted graph $D(G)$, obtained from G by replacing every edge with a new vertex connected to the two vertices originally joined by that edge.
Inputs: The Client inputs a quantum state ρ_{inp} and the classical description of a unitary U as a flow $(f, \preceq)$ on $DT(G)$ and default measurement angles $\{\phi(v)\}_{v \in O^c}$ which, when applied to the dotted graph state $|D(G)\rangle$, perform the desired computation from inputs in I to outputs in O in a fault-tolerant way that can detect or correct up to δ errors. The Server has no input.
Protocol:

1. The Client chooses a trap-colouring for the dotted triple-graph $DT(G)$ according to Definition 3.15. Let D be the set of positions of dummy qubits and for all $v \notin D$, $D(v) := N_G(v) \cap D$. It samples uniformly random $\theta(v) \in_R \Theta$ and $r(v) \in_R \{0, 1\}$ for each computation and trap qubit, uniformly random $a(v) \in_R \{0, 1\}$ for each input qubit and uniformly random $d(v) \in_R \{0, 1\}$ for each dummy qubit. It initialises a string $\mathbf{s}$ to 0 representing the measurement outcomes on G.
2. The Client encrypts its input qubits $\bigotimes_{v \in I} \mathsf{Z}(\theta(v))\mathsf{X}^{a(v)}\rho_{inp}$ and prepares dummy qubits in state $|d(v)\rangle$ and the remaining qubits in state $\bigotimes_{w \in D(w)} \mathsf{Z}^{d(w)} \left|+_{\theta(v)}\right\rangle$. It sends to the Server all the qubits in the order of the labelling of the graph.
3. The Server receives the qubits one-by-one and applies the entangling operations CZ that correspond to the edges of the graph $DT(G)$.
4. For each qubit $v \in O^c$, following the partial order of the flow:
   a) The Client generates $r(v) \in_R \{0, 1\}$, calculates $\delta(v)$ according to Equation 3.16, using the value $\phi'(v) = 0$ for trap and dummy qubits, and sends it to the Server.
   b) The Server measures in the $\{\left|+_{\delta(v)}\right\rangle, \left|-_{\delta(v)}\right\rangle\}$ basis and returns to the Client outcome $b(v)$.
   c) The Client sets the value of $s(v)$ in $\mathbf{s}$ to $b(v) + r(v)$.
5. The Server returns all qubits that correspond to output base-locations to the Client.
6. The Client measures the output trap qubits t with angle $\delta(t) = \theta(t) + r(t)\pi$ to obtain $b(t)$. If there is a trap qubit t (including ones measured previously by the Server) such that $b(t) \neq r(t)$, it outputs Abort and sends it to the Server.
7. Otherwise, the Client applies final Pauli correction $\mathsf{Z}^{\hat{s}^Z(v)}\mathsf{X}^{\hat{s}^X(v)}$ to the output layer green (computation) qubits. It sets these qubits as its output and sends Ok to the Server.

performed on the base graph G of maximum degree c. The VBQC Protocol 5 is ϵ_V-locally-verifiable for the Client (Definition 3.12), where $\epsilon_V = \left(\frac{8}{9}\right)^d$ for $d = \left\lceil \frac{\delta}{2(2c+1)} \right\rceil$.

Note that, compared to the version presented in [78], we explicitly require the Client to inform the Server if it outputs Abort. This is required so that it may satisfy the independent verification property from Definition 3.13, which will allow us later in Chapter 6 to give a new formulation for the security of this protocol by proving its security in the Abstract Cryptography Framework.

3.5.3 The Classical Yao Protocol

Yao's Protocol, the pioneer of Secure Two-Party Computation, allows two Parties, the Garbler and the Evaluator, to compute a joint function on their two classical inputs. The Garbler starts by preparing an encrypted version of the function and then the Evaluator decrypts it using keys that correspond to the two players' inputs, the resulting decrypted value being the final output.

The Original Yao Protocol secure against Honest-but-Curious classical Adversaries has first been described by Yao in the oral presentation for [135], but a rigorous formal proof was only presented

in [89]. It has been proven secure against quantum Adversaries with no superposition access to the honest player in [24] (for a quantum version of IND-CPA that only allows random oracle queries to be in superposition).

We give here only a brief description of the construction and defer to later sections the presentation of the full protocol along with associated correctness and security properties. It uses as primitives a symmetric encryption scheme and 1-out-of-2 Oblivious Transfer.

A symmetric encryption scheme consists of two classical efficiently computable deterministic functions $\mathsf{Enc} : \mathfrak{K} \times \mathfrak{A} \times \mathfrak{M} \to \mathfrak{K} \times \mathfrak{A} \times \mathfrak{C}$ and $\mathsf{Dec} : \mathfrak{K} \times \mathfrak{A} \times \mathfrak{C} \to \mathfrak{K} \times \mathfrak{A} \times \mathfrak{M}$ (where $\mathfrak{K}$ is the set of valid keys, $\mathfrak{A}$ the set of auxiliary inputs, $\mathfrak{M}$ the set of plaintext messages and $\mathfrak{C}$ the set of ciphertexts, which is supposed equal to $\mathfrak{M}$). We suppose that for all $(k, \mathrm{aux}, m) \in \mathfrak{K} \times \mathfrak{A} \times \mathfrak{M}$, we have that $\mathsf{Dec}_k(\mathrm{aux}, \mathsf{Enc}_k(\mathrm{aux}, m)) = m$.

We focus on the case where the Garbler's output to Yao's Protocol is a single bit (and the Evaluator has no output). Suppose that the Garbler and Evaluator have agreed on the binary function to be evaluated $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \longrightarrow \{0,1\}$, with the Garbler's input being $x \in \{0,1\}^{n_X}$ and the Evaluator's input being $y \in \{0,1\}^{n_Y}$.

The protocol works as follows. The Garbler samples keys $\left\{k_0^{G,i}, k_1^{G,i}\right\}_{i\in[n_X]}$ and $\left\{k_0^{E,i}, k_1^{E,i}\right\}_{i\in[n_Y]}$ for the Garbler's and Evaluator's input respectively. To each bit of input correspond two keys, one (lower-indexed with 0) if the player chooses the value 0 for this bit-input and the other if it chooses the value 1. They invoke n_Y instances of a 1-out-of-2 String OT Ideal Functionality, the Evaluator's input (as Receiver of the OT) to these is y_i for $i \in [n_Y]$, while the Garbler inputs (as Sender) the keys $(k_0^{E,i}, k_1^{E,i})$ corresponding to input i of the Evaluator. The Evaluator therefore recovers $k_{y_i}^{E,i}$ at the end of each activation of the OT. The Garbler then sends the keys $\left\{k_{x_i}^{G,i}\right\}_{i\in[n_X]}$ corresponding to its own input along with the garbled circuit GC_f which is constructed as follows.

For a gate computing a two-bit function g, with inputs wires labelled a and b and output wire z, the Garbler first chooses keys $(k_0^a, k_1^a, k_0^b, k_1^b) \in \mathfrak{K}^4$ for the input wires and $k^z \in \{0,1\}$ for the output.[7] Let aux_a and aux_b be two auxiliary values for the encryption scheme. It then iterates over all possible values $\tilde{a}, \tilde{b} \in \{0,1\}$ to compute the garbled table values $E_{\tilde{a},\tilde{b}}^{k^z}$ defined as (with padding length $p = n_M - 1$, where n_M is the bit-length of the messages of the encryption scheme[8]):

$$E_{\tilde{a},\tilde{b}}^{k^z} := \mathsf{Enc}_{k_{\tilde{a}}^a}\Big(\mathrm{aux}_a, \mathsf{Enc}_{k_{\tilde{b}}^b}(\mathrm{aux}_b, g(\tilde{a},\tilde{b}) \oplus k^z \parallel 0^p)\Big) \tag{3.17}$$

The ordered list thus obtained is called the *initial* garbled table. The Garbler then chooses a random permutation $\pi \in \mathfrak{S}_4$ and applies it to this list, yielding the *final* garbled table $GT_g^{(a,b,z)}$. For gates with fan-in l, the only difference is that the number of keys used will be $2l$ (two for each input bit) and the number of values in the garbled table will be 2^l, the rest may be computed in a similar way (by iterating over all possible values of the function's inputs). The keys are always used in a fixed order which is known to both players at time of execution (we suppose for example that, during encryption, all the keys of the Evaluator are applied first, followed by the keys of the Garbler).

Finally, after receiving the keys (through the OT protocols for its own, and via direct communication for the Garbler's) and garbled table, the Evaluator uses them to decrypt sequentially each entry of the

[7] The value k^z is used to One-Time-Pad the outputs, preserving security for the Garbler after decryption as only one value from the garbled table can be decrypted correctly.

[8] The padding enforces the verifiable and elusive range property required in the original Yao Protocol [135, 89].

table and considers it a success if the last p bits are equal to 0. It then returns the decrypted value to the Garbler.

3.5.4 Universal Thresholdiser

A *Universal Thresholdiser* (or UT) can be seen as a combination of *Fully-Homomorphic Encryption* (FHE), which allows for computations to be performed on encrypted inputs, and a *Secret Sharing Scheme*, which is used to distribute a secret to multiple parties and later reconstruct it if enough players divulge their received shares. It enables a certain number of parties to encrypt separately values using a joint encryption key, share these values with the other parties, who can all perform a pre-determined computation and broadcast the output (corresponding to its share of the final secret). If a sufficient number of parties perform these operations, then each parties having received these computed shares can decrypt the result of the computation. The definitions of this functionality are mostly taken from [16], but they must be adapted in some places to fit the use that will be made of the UT later in the thesis. Most notably, we adapt the security definition to fit the AC framework described in Section 3.3.2 and require it to be secure against malicious quantum Adversaries. We also correct their definition of robustness of a UT scheme. Modifications straying from the original presentation of [16] are clearly marked. Consequently, this means that it is not clear whether the instantiation of this scheme by [16] fulfils the updated definitions. This does not affect the results presented in our work since we only suppose the existence of a quantum-secure Classical SMPC, a Universal Thresholdiser being only one of the possible ways to achieve this functionality. The reason why we opt for this construction rather than another is that a UT based on FHE allows the reuse of intermediate computations, which in turn allows us to fine-grain the description of the functions that we need to implement using this tool. The final goal is to showcase the simplicity of those functions compared to other protocols to seek to construct the same MPQC functionality as we do in Chapter 6.

We use the definition of a Universal Thresholdiser from [16]. We consider only *Threshold Access Structures*, essentially meaning that any group of parties of size larger or equal to a certain threshold can gain access to the decrypted values (for more detail see Section 4 of [16]). Even more specifically, we will only require the case where $t = N$, meaning that all parties must collaborate to decrypt values.

Definition 3.17 (Universal Thresholdiser)**.** *A Universal Thresholdiser (or UT) is a tuple of efficient classical protocols* $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verif}, \mathsf{Combine})$ *defined as follows:*

- $\mathsf{Setup}(1^\eta, 1^N, 1^t, 1^d, \mathbf{pub}, x) \to \left(\mathbf{pp}, \{sk_i\}_{i\in[N]}\right)$*: It is a multi-party protocol that takes as input a security parameter* η*, the number of parties* N*, a threshold* t*, a bound on the binary circuit depth* d *and an input* x *and outputs public parameters* $\mathbf{pp}$ *(it contains a public key* pk *and encryptions* c_p *and* c_x *of some public parameters and the input* x *respectively), along with private parameters for each party* sk_i *(including a signature key). We suppose that this step can be decomposed into two parts: first the secret parameters are created and shared using a protocol* $\widetilde{\mathsf{Setup}}(1^\eta, 1^N, 1^t, 1^d) \to \left(pk, c_p, \{sk_i\}_{i\in[N]}\right)$*, then all parties encrypt their inputs and the public parameters separately under the public key* pk *and share them publicly, thereby creating* c_p *and* c_x*.*
- $\mathsf{Eval}(\mathbf{pp}, C, sk_i) \to y_i$*: It is an algorithm (performed by each player), taking as input the public parameters, the description of a binary circuit* C *and the secret parameter of party* i *and outputting*

a partial evaluation y_i (it is called partial since it is impossible to find the correct evaluation without the other partial values).

- $\mathsf{Verif}(\mathbf{pp}, C, y_i) \to 0/1$*: The verification algorithm takes as input a partial evaluation and a binary circuit (along with the public parameters) and outputs* 1 *if the partial evaluation is coherent with the circuit description (and* 0 *otherwise).*
- $\mathsf{Combine}(\mathbf{pp}, \{y_i\}_{i\in S}) \to y$*: For a subset $S \subseteq [N]$, if $\#S \geq t$, the combining protocol outputs the final evaluation y (as plain-text).*

Resource 15 corresponds to the first step of the setup protocol of the Universal Thresholdiser defined above (it has the same number of interfaces N as there are Clients). It chooses the public and secret keys and then only encrypts under public-key pk the known (clear-text) public parameters **pub** (otherwise each Client would have to encrypt it separately and later test that it has been done correctly by the other Clients). We suppose that this resource is available to the Clients.

Resource 15 Secure UT Key Setup

Public Information: Number of parties N, threshold t, upper bound on the total depth d of the classical circuit, security parameter η, public parameters **pub**.
Inputs: All N parties input the dummy input λ.
Computation by the Resource: It runs the protocol $\widetilde{\mathsf{Setup}}(1^\eta, 1^N, 1^t, 1^d) \to \left(pk, \{sk_i\}_{i\in[N]}\right)$ internally to get the secret parameters. It then encrypts the public parameters **pub** under the public key pk, obtaining cypher-text c_p, and sends (pk, c_p, sk_i) to party P_i.

The protocol $\mathsf{Setup}(1^\eta, 1^N, 1^t, 1^d, \mathbf{pub}, x)$ then consists of first calling the Ideal Resource above and then having each Client encrypt its own private inputs and sharing the encrypted values with the other Clients, yielding the global public parameters **pp** (containing the public key, the encrypted publicly known values and encrypted private values).

We can now define the properties satisfied by a Universal Thresholdiser. It is said to be compact if the size of the partial evaluations does not blow up, regardless of the classical circuit that is being computed (for bounded-depth circuits).

Definition 3.18 (UT Compactness)**.** *A UT scheme for N parties is said to be compact if there exists a polynomial p such that for all η, depth-bounds d, bounded-depth binary circuits C, inputs x, $\left(\mathbf{pp}, \{sk_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\eta, 1^N, 1^t, 1^d, \mathbf{pub}, x)$ and $y_i \leftarrow \mathsf{Eval}(\mathbf{pp}, C, sk_i)$, we have $\#y_i \leq p(N, \eta, d)$.*

The Universal Thresholdiser is said to be correct if the probability that honestly generated partial evaluations do not recombine correctly is negligible and the partial evaluations always pass verification (this combines two definition from [16] into one).

Definition 3.19 (UT Correctness)**.** *A UT scheme for N parties is said to be evaluation-correct if there exists a negligible function $\epsilon_{UT,cor}(\eta)$ such that for all η, depth-bounds d, bounded-depth binary circuits C, inputs x, sets of parties $S \subseteq [N]$ with $\#S \geq t$, $\left(\mathbf{pp}, \{sk_i\}_{i\in[N]}\right) \leftarrow \mathsf{Setup}(1^\eta, 1^N, 1^t, 1^d, \mathbf{pub}, x)$ and $y_i \leftarrow \mathsf{Eval}(\mathbf{pp}, C, sk_i)$, we have:*

$$\Pr\left[\mathsf{Combine}\left(\mathbf{pp}, \{y_i\}_{i\in S}\right) = C(x)\right] = 1 - \epsilon_{UT,cor}(\eta) \tag{3.18}$$

It is further said to be verification-correct if for all $i \in [N]$:

$$\Pr\big[\mathsf{Verif}(\mathbf{pp}, C, y_i) = 1\big] = 1 \tag{3.19}$$

It is said to be robust if no prover can convince the verification algorithm to accept an incorrectly generated partial evaluation with non-negligible probability. Here we consider QPT Adversaries instead of PPT Adversaries in [16]. We believe that the scheme presented in [16] is secure against quantum adversaries as well since it is based on the Learning-With-Error assumption (or LWE). However the paper lacks a formal proof against QPT Adversaries and proofs of security against quantum adversaries are notoriously more difficult than in the classical setting, especially when using zero-knowledge constructions, as demonstrated in [7]. [9]

Definition 3.20 (UT Robustness)**.** *A UT scheme for N parties is said to be robust if there exists a negligible function $\epsilon_{UT,ver}(\eta)$ such that for all η, depth-bounds d and all* QPT *Adversaries $\mathcal{A}$, we have:*

$$\Pr\big[b = 1 \mid b \leftarrow \mathsf{Exp}_{\mathcal{A},\mathsf{Rob}}(1^\eta, 1^d)\big] = \epsilon_{UT,ver}(\eta) \tag{3.20}$$

where $\mathsf{Exp}_{\mathcal{A},\mathsf{Rob}}$ is the following experiment:

1. *The Adversary takes as input $(1^\eta, 1^d)$ and sends a threshold t, parameters* **pub** *and an input x to the Challenger.*
2. *The Challenger runs $\Big(\mathbf{pp}, \{sk_i\}_{i\in[N]}\Big) \leftarrow \mathsf{Setup}(1^\eta, 1^N, 1^t, 1^d, \mathbf{pub}, x)$ and sends $\Big(\mathbf{pp}, \{sk_i\}_{i\in[N]}\Big)$ to the Adversary.*
3. *The Adversary outputs $(i, \hat{y}_i, C)$ with C being the description of a classical circuit and $i \in [N]$.*
4. *The Challenger outputs 1 if $\mathsf{Verif}(\mathbf{pp}, C, \hat{y}_i) = 1$ and for all possible values $y_i \leftarrow \mathsf{Eval}(\mathbf{pp}, C, sk_i)$, we have that $y_i \neq \hat{y}_i$.*

The whole scheme is said to be secure if (i) no coalition of malicious parties of size lower than t can learn anything about the input or output and (ii) if given the partial evaluations of honest parties, they cannot learn anything from the input apart from what can be inferred from the output. This is defined in a simulation-based way in [16] and we update it to quantum security by changing PPT machines into QPT machines and redefining it in the Abstract Cryptography setting. We start by defining the following Threshold SMPC Protocol 6 using the Secure UT Key Setup Resource 15.

For the UT to be considered secure, Protocol 6 should emulate the Secure Classical Multi-Party Computation Resource 10 for any coalition of less than t malicious parties. This definition of security implies the one from [16] but the converse is not true (since quantum adversaries are not taken into account). [10]

[9] There is also an imprecision in the robustness game as defined in [16], as it implies that the Eval algorithm is deterministic, while their construction is not (in particular the evaluation samples a smudging error value at random). We fix this by making the Challenger more powerful (it can compute all values of the evaluation function for a given binary circuit and input), but this does not impact the security of the scheme in any way. Fixing the game in a different way (giving the Adversary access to an oracle for example) would be much more troublesome (especially in the quantum case).

[10] In Definition 7.5 from [16], algorithm $\Big(\mathbf{pp}, \{sk_i\}_{i\in[t-1]}, \mathbf{st}\Big) \leftarrow \mathcal{S}_1(1^\eta, 1^d, t, \mathbf{pub})$ impersonates the setup procedure without knowing the input of honest players, while algorithm $\mathcal{S}_2(\mathbf{pp}, C, C(x), S, \mathbf{st})$, with $S \subseteq [N]$ uses the state **st** to

Protocol 6 Threshold SMPC Protocol

Public Information: The description of a classical joint circuit C, threshold t, parameters **pub**.
Inputs: The N Parties each input a value x_i.
Outputs: Each party receives the result $C(\mathbf{pub}, x_1, \ldots, x_N)$ or Abort.
Protocol: Computation by Party i:

1. Party i sends the dummy input λ to the Secure UT Key Setup Resource 15 and receives the initialisation values (pk, c_p, sk_i).
2. It generates an encryption of x_i using the public key pk and broadcasts these encryptions to all other Parties.
3. It receives corresponding encryptions from the other Parties, uses Eval to apply the classical circuit C, generating a partial evaluation y_i and broadcasting these partial evaluations to all Parties.
4. It receives partial evaluations from other Parties $\hat{y}_j$ (for $j \neq i$), applies the verification algorithm Verif on all of them and outputs Abort if less than t verifications succeed. Otherwise it uses the algorithm Combine on the evaluations that passed verification to recover the output y.

Definition 3.21 (Universal Thresholdiser Security)**.** *A UT scheme for N parties is said to be secure if there exists a negligible function $\epsilon_{UT}(\eta)$ such that for all η, depth-bounds d, bounded-depth classical circuits C and thresholds t, Protocol 6 $\epsilon_{UT}(\eta)$-computationally-constructs the Threshold Verifiable Multi-Party Classical Computation Resource 10 from the Secure UT Key Setup Resource 15 against all* QPT *Adversaries $\mathcal{A}$ controlling a coalition of $c < t$ corrupted Parties.*

Finally, we suppose that partial evaluations can be reused in later evaluations as long as the total depth of the classical circuit does not exceed d (this is usually the case with FHE-based schemes but is not explicitly allowed in the definition of the Universal Thresholdiser).

create "fake" partial evaluations that both pass verification and output the correct result of the computation for the chosen classical circuit (it knows only the circuit and the output of the circuit). These algorithms are classical in [16] but may need to be quantum against QPT Adversaries as there is no guarantee that a classical Simulator exists in general against a quantum Adversary, even for classical protocols.

Chapter 4

Quantum Protocol Compiler: Boosting Security from Semi-Malicious to Malicious Adversaries

4.1 Motivation and Overview of Results

4.1.1 Weaker Adversaries, Simpler Protocols

Numerous types of different adversaries can be defined in the context of secure protocols. These not only differ by the computational power at their disposal, but also in the ways in which they may choose to deviate from the specifications given by the protocol's description. By definition, Malicious Adversaries can cause more damage the Honest-but-Curious ones since they can perform unauthorised computation. As a general principle and for any given task, it is harder to design protocols the resist against parties that deviate more since they are by definition less constrained by the protocol's rules that bind honest players. This reasoning is also applicable when considering the adversary's computational power, explaining intuitively why there are comparatively fewer information-theoretically secure protocols. Impossibility results demonstrate the validity of this idea, as even quantum powers are not sufficient to provide unconditionally-secure Bit Commitment for instance [95, 100, 93].

It is possible to look at this issue the other way around, by first building a simple protocol that is secure against weak adversaries and finding later ways to force these adversaries to behave honestly, therefore reducing the problem to simpler Honest-but-Curious behaviour analysis. Tools for achieving such security boosting include for example the universal GMW Compiler from [61]. In the classical case it forces the sender of a message to prove that this message has been generated honestly, meaning that there exist secret parameters which are consistent with this message and the previous transcript being generated by an honest player. This is done by reducing the validity of the statement "the parties are acting honestly" to a valid solution of a hard instance of an NP-Complete problem (constraint

satisfiability) and requires each party to use generic Zero Knowledge Proofs to prove that this statement is satisfied at each step of the protocol (additionally the proof should reveal nothing apart from the fact that it is correct). Unfortunately this comes at the cost of efficiency and, while it does run in polynomial time, it is mostly considered a theoretical result with no hope of being implemented in this generic form.

Another technique which has shown good results with regard to efficiency is Cut-And-Choose. It is used to convince a player in a protocol that a given message has been produced according to its correct specification (in that sense its purpose is similar to the GMW Compiler above). In order for the player receiving the message (Receiver) to verify the honesty of the prepared message, the player sending the message (Sender) creates multiple independently generated copies of the message. The Receiver then chooses which ones (the *check sets*) they will check for consistency and request additional information from the Sender in the form of a corresponding "proof of honesty" for these state-message pairs. If the checks pass and additional precautions are taken, the Receiver is confident that with high probability the remaining states (the *evaluation sets*) were also constructed correctly and can be used for the computation. The Sender's security is preserved if the CC procedure does not reveal more information about the sent message apart from the fact that it is correct. This technique has been part of cryptographic "folklore" for a long time, but has never been properly formalised by itself, even in the classical case. We extend this technique to the quantum case, yielding the Quantum Cut-and-Choose (Q-CC), by performing it on $(|\psi_{sk}\rangle, m_{sk})$ where $|\psi_{sk}\rangle$ is a quantum state and m_{sk} represents additional classical information, both depending on some secret sk. The proof of honesty associated to such messages and states will be denoted $proof_{sk}$.

Our work in the quantum setting focuses on the case where there are s sets generated in total, $s-1$ *check sets* and 1 *evaluation set*, thus yielding inverse-polynomial security. In the classical setting, the probability of cheating and not getting caught can be made negligible by using $s/2$ check sets and $s/2$ evaluation sets. We extend this framework by considering a more general protocol where a fraction κ of sets are checked, with the guarantee that at most a fraction α of the unchecked sets are corrupted. If using more than 1 evaluation set, the analysis in the quantum case is made even more complex due to coherent (entangled) attacks across repeated copies and we leave open the question regarding the possibility of achieving security in this setting.

4.1.2 Our Contribution

We give now a brief overview of the results in this chapter. The links between the sections regarding the Quantum Cut-and-Choose are given in Figure 4.1. Section 4.3 is mostly stand-alone and its structure is given in Figure 4.2. The numbering for definitions, functionalities, protocols, theorems both refer to that diagram and the main text.

Section 4.2: Inverse-Polynomial Quantum Cut-and-Choose. The Cut-and-Choose technique is mainly used to prove that a message has been generated as promised, when revealing some of the secret parameters used to generate it would compromise the security of the overall protocol in which this message is used. This is the first time that the Cut-and-Choose technique is formalised by itself instead of serving as a subroutine for a larger protocol. This is crucial as using it as it was done up to now has led to attacks [88, 80]. We therefore define the purpose of the Quantum Cut-and-Choose Protocol as the emulation of an Ideal Functionality called Send Blind Correct State for a Sender and a

Receiver, which guarantees that the Receiver receives a quantum state and a classical message satisfying certain properties (a Malicious Sender can however choose which state it is, as long as said properties are satisfied). It does not however reveal to the Receiver more information that would allow it to verify the sent state and message by itself. This is formalised as Ideal Functionality 16. Of note is that the quantum case that we present is a strict generalisation of the classical case, which can be seen by either constraining the set of states to computational basis states or even remove the state altogether.

However, this Ideal Functionality cannot be applied to all states and classical messages. First of all, there should exist a proof that allows the person receiving a state and message to verify that it has in fact been generated correctly. Given this proof, the Receiver should be able to extract from the message a classical description of the state, which allows it to measure it and test its correctness (see the Extractability Definition 4.2). On the other hand, the fact that the proof of correctness has been given for multiple states and classical messages should not give any information about any subsequently generated state and message, not even whether it was generated using the same secret parameters (as captured by the Key-Indistinguishability Definition 4.5). Finally, the Cut-and-Choose procedure is never useful in itself but as a subroutine in a global protocol. This protocol may require that some parts of the proof of the message be revealed. We model these as leaks and define the maximal tolerated leakage of the proof (which may be composed of multiple parts) as the maximum amount of information that can be leaked about the proof without breaking the definitions above. These leaks should on the other hand be verifiable by a Receiver in the sense that it can test whether or not an information is a leak of the proof for a given message (Leak-Verifiable Definition 4.3). The states and messages verifying these properties are called Cut-and-Choosable (or CC-able, Definition 4.1).

This allows us to formalise the Quantum Cut-and-Choose Protocol 7 where the Sender produces s independently generated copies of the message and state, the Receiver chooses one of them and asks for the proofs of the other ones in order to test them. If the Sender sends the proofs and the tests pass, the Receiver is convinced that the remaining copy would have also passed the test if it had been given the proof. This is captured by the following informal Theorem 4.1:

Theorem 4.1 (Security of the Quantum Cut-and-Choose Protocol, Informal)**.** *If the states and messages are CC-able, the Q-CC Protocol 7 $1/\sqrt{s}+\epsilon$-securely emulates the Send Blind State Ideal Functionality 16 in the Stand-Alone Framework of [66] against a Malicious Sender, for a negligible ϵ. The security bound is negligible for an adversarial Receiver (and directly linked to the key-indistinguishability of the states).*

A key technical obstruction when using classical techniques for boosting the security of quantum protocols is that in general it is *not* possible to use the rewinding method during the simulation for proving security against *quantum* Adversaries. Some classically-secure protocols using this technique have been proven insecure in the quantum case in [7], which shows attacks relative to a specific oracle. There are a few known cases where rewinding *can* be used on such Adversaries, in particular Watrous' oblivious rewinding [130] and Unruh's special rewinding [127, 129]. We adapt and use both methods in different places in order to construct the Simulators proving the security of our protocol. This is one of the few protocols in which quantum rewinding is explicitly used and therefore provides a good example of how and when these techniques should be applied. The proof of Theorem 4.1 informally works as follows:

Proof of Theorem 4.1 (Sketch) The Theorem is proven by considering the two malicious cases separately.

Simulator against Malicious Receiver. It receives from the Ideal Functionality the message and state that it needs to make the Receiver accept as valid (however it does not know the proof for this state). It generates on its own $s-1$ states, messages and proofs and sends all states and messages to the Receiver, applying a permutation to hide the position of the ideal state. The Receiver chooses one of the states and requests the proof for the remaining ones. If it has chosen the ideal state, then the Simulator can go through and send the proofs for the other states (the ones that it has generated). The Receiver can check these states and due to the key-indistinguishability cannot deduce that the remaining (ideal) state has not been generated in a similar manner (using the same key). However, if the Receiver has not chosen the ideal state, the Simulator cannot continue its simulation and is forced to rewind the Adversary, choose a different permutation and repeat until the ideal state has been chosen. This is made possible by using the *oblivious rewinding technique* of [130] as the probability of succeeding in the simulation is constant and independent of the internal state of the Adversary (the simulation succeeds if the Adversary picks the ideal state as unchecked, which happens with probability $1/s$).

Simulator against Malicious Sender. The Simulator against a Malicious Sender receives all the states and messages from the Adversary. Its task is to send a message and a proof to the Ideal Functionality. These elements should correspond to the state and message and state chosen as unchecked. It therefore faces a seemingly impossible task: the Adversary will not send the proof for the unchecked state and message, but the Simulator needs to forward precisely this proof to the Ideal Functionality. This is solved by making use of the *special rewinding technique* of [127, 129]. The Simulator runs the Adversary and picks one state and message as unchecked, receives the corresponding proofs. It then rewinds the Adversary and chooses a different state and message as unchecked, again receiving the proofs. It now has all the proofs and can send the correct one to the Ideal Functionality. Since, in the Stand-Alone Model of security, each classical interaction is modelled as a measurement, in general this procedure is not possible because the first classical interaction will have disturbed the state of the Adversary: being able to repeat this with no restrictions is equivalent to cloning the Adversary's internal state, which is impossible in the quantum case. This forces us to put an additional constraint on the CC-able states and messages: measuring the quantum register containing the message should be equivalent to having measured both the message and the proof (a stronger version would be to require each correct proof to be uniquely determined by the corresponding message). This is captured by the Collapsing Proofs Definition 4.4 and implies that the internal state of the Adversary is minimally disturbed by revealing the proofs, allowing the Simulator to rewind it.[1] We also give in Section 4.2.5.2 a new characterisation of the distance between the ideal and real states (ie. with and without rewinding), while previous works [127, 129] are only concerned with linking the probability that a player succeeds twice in a game to the probability that it succeeds once.

■

Section 4.3: Exponentially-Secure Fraction Classical Cut-and-Choose. In this section, which is mostly independent of the rest of this chapter, we show a formalisation of the Classical Cut-and-Choose

[1] Notice that the previous rewinding technique of [130] does not work as it does not allow the Simulator to keep information between rewinds beyond the fact that it has succeeded or failed the simulation, hence the name *oblivious* rewinding.

technique where a fraction of the sets are checked and the rest are transmitted. The ideal scenario in this setting guarantees that at most a fixed fraction of the forwarded sets have been corrupted. As above, the Receiver does not receive proofs that the received messages are correct and does not know even which ones would pass a test. This version of Cut-and-Choose is useful in the case where an outer protocol is capable of naturally recovering from a fraction of incorrect results by using some form of classical error-correction. The most often used technique is to perform a majority vote over the remaining messages, in which case the corrupted fraction can be as high as $1/2$ without impacting the final result.

In order to demonstrate an alternative to the proofs of security of CC presented in the previous section, we define the ideal scenario as the Send Blind Correct Fraction of Messages Ideal Resource 18 in the fully composable Abstract Cryptography framework of [99]. The number of transmitted messages and the maximal fraction of incorrect ones are both hard-coded inside this Resource in the sense that it expects a given number of message from the Sender's interface if malicious, and correct proofs for a given fraction of these messages. Rewinding is forbidden in such composable frameworks and we therefore must adapt the conditions for applying the Fraction Classical Cut-and-Choose Protocol so that simulation remains possible. In addition to the Key-Indistinguishability property which is still required, we use a setup assumption which gives the Simulator access to a trapdoor with which to open any message (Trapdoor Extractability, Definition 4.10). The setups used by the protocol and the simulation must then be indistinguishable, a property captured by the Indistinguishable Dual-Mode Setup Definition 4.9. The *classical Sender and Receiver algorithms* that satisfy these three properties are then called F-CC-able (Definition 4.8). Note that we do not deal with leaks here for simplicity's sake.

Given these definitions and after the trusted setup has been executed, the Fraction Classical Cut-and-Choose 8 has the Sender produce s messages, the Receiver chooses a subset to check and receives the proofs, aborting if any message fails the test. In order for the protocol to be simulatable in polynomial-time, choosing the subset of checked sets is done via a call to a Coin-Tossing Resource. The trusted setup is modelled by a Common Random String Resource (CRS). The security is captured by the following informal Theorem 4.2:

Theorem 4.2 (Security of the Fraction Classical Cut-and-Choose, Informal)**.** *If the messages are F-CC-able, the FC-CC Protocol 8 ϵ-securely constructs the Send Blind Correct Fraction of Messages Resource 16 in the Abstract Cryptography of [99] against Malicious Receiver and Sender, for a negligible security bound ϵ.*

Proof of Theorem 4.2 (Sketch) *Simulator against Malicious Receiver.* It receives from the Ideal Resource f messages that it needs to make the Receiver accept, without the corresponding proofs. It generates on its own $k = s - f$ messages and proofs and sends all states and messages to the Receiver after mixing them together. It impersonates the Coin-Tossing Resource and sends to the Receiver the check subset corresponding to the simulator-generated ones, along with the associated proofs. These messages pass the test and the Distinguisher is fooled by the Key-Indistinguishability property.

Simulator against Malicious Sender. The Simulator must send f messages to the Ideal Resource, along with proofs for at least h of them. It impersonates the CRS Resource performing the trusted setup and generates a trapdoor for itself. The Simulator then receives s messages from the Adversary. It sends a random check subset to the Sender via the Coin-Tossing Resource and receives proofs in return. If all the checks pass, it uses the trapdoor on the remaining messages to extract their proofs. If less than h

proofs are valid, the simulation fails as the Adversary has successfully attacked the protocol. Otherwise the Simulator forwards these to the Ideal Resource.

■

Beyond this formalisation, we derive for each maximal fraction of corrupted sets the optimal fraction of tests that need to be performed to minimise the security bound of our protocol. Rather counter-intuitively, for the case mentioned above where a majority vote is applied in the end, this gives an optimal fraction of tested sets of 3/5, as opposed to the usually used 1/2. All protocols relying on this technique can automatically benefit from improved bounds by simply switching to this new value. Finally we prove that improving the error-correcting procedure of the outer protocol also improves the security bound, implying a lower number of sets generated during the FC-CC Protocol for a fixed security level.

Section 4.4: Protocol Compiler. While the Stand-Alone Security Framework of [66] ensures that any protocol that is proven secure can also be sequentially composed with any other, we also give a Protocol Compiler which uses the Quantum Cut-and-Choose technique to boost the security of protocols satisfying certain constraints. This is again completely orthogonal to existing approaches as they study the applicability of this procedure on a case-by-case basis. The Compiler is required here since we are effectively combining different executions of a protocol, each execution will be linked to a set that will be used in the Q-CC Protocol. The sequential composability property of the Stand-Alone Framework only guarantees that using Q-CC as a sub-protocol in a larger protocol preserves security, not that it can be used to merge executions of other protocols to amplify the security.

We start by introducing a new type of Adversary called Semi-Malicious since they may deviate in all steps of the protocol apart from a certain number of *classical* message which they generate in a manner which is indistinguishable from an honest player. Proving security against these Adversaries is much easier than against fully Malicious ones. Our Compiler takes care of automatically upgrading the security of these protocols by enforcing the honest generation of these messages even against a fully Malicious Adversary. This can be used for example to secure future messages in the protocol which rely on the state generated through the Quantum Cut-and-Choose procedure. These may be instructions regarding how the state should be used later on in a computation, whether it is the description of measurement or unitaries to be applied which may depend on future interactions, or classical post-processing of quantum computations. It allows the protocol to thwart malicious strategies in which the Adversary is forced to produce a correct state by the Quantum Cut-and-Choose procedure but can then adaptively cheat in by giving false instructions (in which case certifying the correctness of the state is rendered meaningless).

The Compiler essentially works by making the Sender pre-compute all possible values of the message to be secured (those for which the Semi-Malicious Adversary is honest) and commit to these pre-computed messages. These commitments can then be tested by the Receiver (the opening value serves as the proof). The protocol is then run s times until the Q-CC step, then the Q-CC procedure takes place and the final steps of the protocol are only run on the unchecked instance. There is of course a trade-off between the complexity of pre-computing a large number of messages and the power of the Semi-Malicious Adversary. We prove that the Compiler preserves all correctness (Lemma 4.6), verifiability (Lemma 4.7) and security properties (Lemmata 4.9 and 4.8) of the original protocol and effectively boosts the security of the protocol from Semi-Malicious Senders to fully Malicious ones (Lemma 4.10).

Naturally, not all protocols can undergo this transformation and we give a set of conditions for it to be applicable. The non-trivial nature of these restrictions, even in the case of 1-out-of-s Q-CC (which is usually considered the "simple" one), shows that this technique is still widely understudied with regard to its general applicability.

Section 4.5: Compiler Application to Secure Two-Party Computation. We then showcase the capabilities of the aforementioned Q-CC technique and more specifically the Compiler by boosting the security of a protocol for Two-Party Quantum Computation with classical inputs and quantum outputs based on the VBQC Protocol of [78] and the QYao Protocol of [77].

We prove that this protocol is computationally-secure against a Semi-Malicious Client (Lemma 4.11). The Server on the other hand can already be fully Malicious and computationally unbounded (Lemma 4.14). Since it also fulfils all requirements for the application of the Compiler (Theorem 4.7), the compiled protocol is therefore also unconditionally secure against a Malicious Server and computationally-secure against a Malicious Client (with an inverse-polynomial security bound due to the use of Q-CC). The proof of security against Semi-Malicious Adversaries is particularly simple (close to the Honest-but-Curious setting), which goes to demonstrate the power of our Compiler.

Our protocol crucially differs from [44, 45] in a number of points (other than using the Quantum Cut-and-Choose technique): (i) there is only two rounds of quantum communication,[2] one of which may be done offline (i.e. before choosing the inputs to the computation); (ii) only one party (called the Server) needs involved quantum technological abilities, while the other (called Client) only needs to prepare single qubits offline and decode Quantum One-Time Pads in the case of quantum outputs; (iii) minimal classical cryptographic primitives are required, namely 1-out-of-2 String Oblivious Transfers for input bits and quantum-safe Bit Commitments (compared to a full classical Secure Two-Party Computation primitive); (iv) these primitives can be readily instantiated to preserve the statistical security of the Client given by the protocol of [78].

Related Works. The field of classical Secure Two (and Multi) Party Computation started with Yao's paper [135], which was proven secure against Malicious Adversaries in [61] using generic Zero-Knowledge Proofs and in [88] with the Cut-and-Choose technique. Covert Adversaries were introduced in [11] where again the Cut-and-Choose technique was used to achieve an even more efficient protocol. Yao's Protocol has been used for a number of other functionalities, such as constructing non-interactive verifiable computing [55].

In the early days of quantum computation, researchers believed that quantum properties could lead to a breakthrough and achieve, with unconditional security, several (classical) multi-party cryptographic primitives. However a series of no-go theorems, first proving that Bit Commitment is impossible [95, 100], then Oblivious Transfer [93]. In the end, [121] showed that any non-trivial ideally-realised functionality leaks some information to Adversaries. Since then, it is established that any such protocol is either only computationally secure or requires the existence of certain (quantum secure) simple cryptographic primitives. In the fully composable setting, the question of completeness and feasibility of functionalities using quantum protocols, both in the computational and statistical case, has been closed by [126, 49, 42].

[2] Only one in the case of classical outputs.

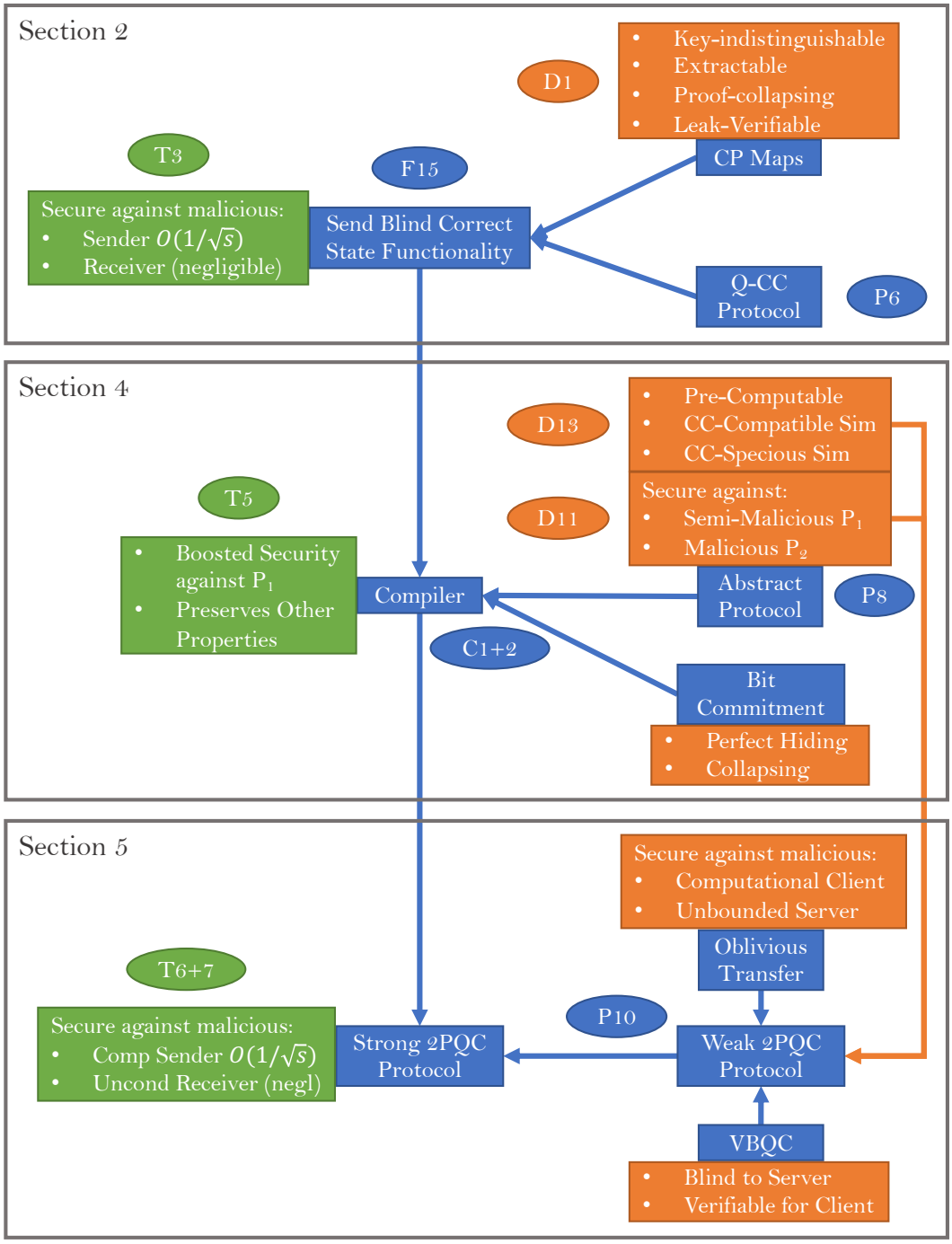


Figure 4.1: Links between Sections 4.2, 4.4 and 4.5 for the Quantum Cut-and-Choose. Protocols (P), Ideal Functionalities (F) and the Compiler (C) are represented in blue, Definitions (D) are coloured orange and properties of protocols given by Theorems (T) are in green. Only new concepts are numbered.

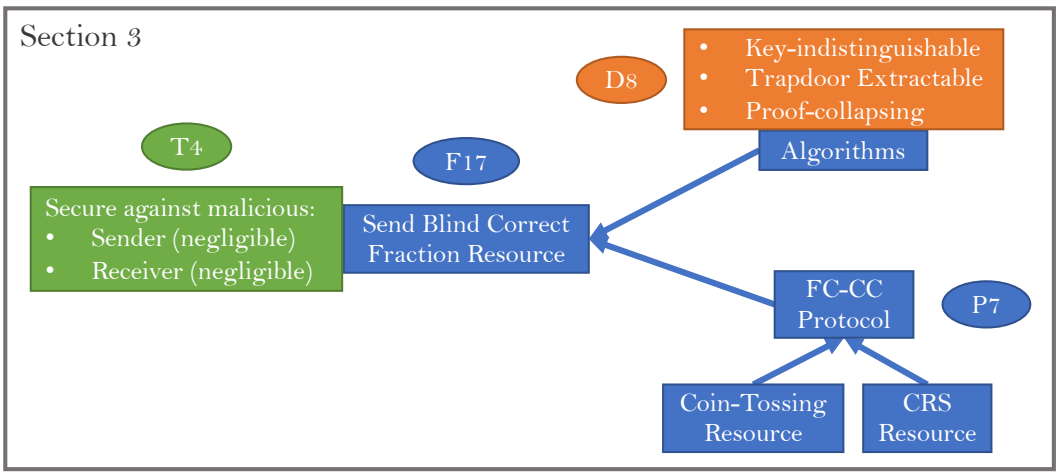


Figure 4.2: Structure of Section 4.3 for the Fraction Classical Cut-and-Choose. The same notations are used as in the previous figure.

Closely related is the question of what assumptions are required if one wants to perform a secure *quantum* computation involving multiple parties. The case of 2PQC was considered in [44] for Quantum Honest-but-Curious and in [45] for Malicious Adversaries. Both of these required a fully-general classical Secure Two-Party Computation primitive to construct the protocol. The case of multiple parties was addressed in [14, 30] where an honest majority is required. More recently, [40] extends [45] to multiple parties while tolerating a dishonest majority.

The 2PQC Protocol of Section 4.5 uses as basis the Measurement-Based Quantum Computation model (MBQC) [118], and more precisely the Universal Blind Quantum Computation Protocol [21] and its verifiable variant [53, 78]. Based on these protocols, [77] gives a 2PQC Protocol against weak Specious Adversaries, while Secure Multi-Party Computation is addressed in [76], again in a restricted setting (non-collusion restriction between a subset of the parties). However, blind and verifiable protocols exist also in the teleportation model [20] or measurement-only model [67] and the question of the existence of 2PQC or MPQC protocols in those frameworks could be similarly explored.

Regarding the problem of proving a player's honesty (or at least mitigating the effects of dishonest behaviour), in Appendix D of [45] they present a technique which appears similar to Q-CC at first glance: to certify the correctness of the magic states $|+_{\pi/4}\rangle$ used during the protocol, one party sends many to the other which tests a constant fraction of them and, if the checks pass, uses magic states distillation on the remaining ones to produce a state exponentially close to a magic state. The main difference is that the receiver knows which state it is supposed to receive whereas in our scheme the remaining state can be unknown. It is unclear whether a similar quantum post-processing would work with unknown states (even if restricted to a known subset of states).

A tightly-connected yet opposite issue is to prove knowledge about a quantum state to a party which possesses only a single copy of this state. This has been studied in [1] where they show that such a task is impossible without leaking information to the party in possession of the state. They prove the security

of a relativistic protocol that works for qudits[3] of large dimension, secure against a limited Sender and a unrestricted Receiver. We show here that a similar task is achievable if the Sender is allowed to make multiple independently generated versions of states that all satisfy a given property.

Remark on Notations. The Semi-Malicious Adversary that is defined later in this work is not related to other definitions of Semi-Malicious Adversaries. Other works starting with [10] consider an Adversary to be Semi-Malicious if they behave as an honest participant but may sample their randomness from different, potentially adversarially chosen, distribution. Our work however defines Semi-Malicious Adversaries as being able to deviate arbitrarily but only for a subset of the steps of the protocol. Furthermore, the Cut-and-Choose Protocol that we discuss here is not to be confused with the Cut-and-Choose primitive that appears in [49, 42]. There, one player inputs a bit x and the other chooses using a bit c whether to receive the bit x or pass (receiving nothing in that case), while the first player gets the choice bit c.

4.2 Inverse-Polynomial Quantum Cut-and-Choose

We start by defining formally the various objects and algorithms that are required for performing a Q-CC Protocol, then define properties that restrict the behaviour of these elements so as to guarantee the correctness and security of the Protocol. The Q-CC Protocol and the Send-Blind-Correct-State Ideal Functionality that it emulates are presented next, followed finally by the proof of inverse-polynomial security against Malicious Sender and full security against Malicious Receiver. The security notion used in this chapter is the Stand-Alone Model of [66] described in Section 3.3.1 and more specifically its two-party setting. These two parties are noted S/P_1 (Sender/Client) and R/P_2 (Receiver/Server).

4.2.1 Formalising the Moving Parts of Quantum Cut-and-Choose

Let $\mathscr{C} \in \{0,1\}^*$ be a public string, with $\#\mathscr{C} = poly(\eta)$ ($\mathscr{C}$ may for example describe the *outer protocol* or computation[4] that both players would like to perform later on the state, or any other public information shared by both players), $D_{sk}(\eta, \mathscr{C})$ a publicly known secret key distribution and $\mathbf{I}$ the classical input space. The distinction between the secret key and the input is that the first is only used in the Q-CC Protocol while the second one may be reused in the outer protocol as well (see Definition 4.5 below).

Let $S_{\mathscr{C}}$ and $R_{\mathscr{C}}$ be QPT-machines (or equivalently, efficient CP-maps) corresponding to respectively the state generation (or Sender) and verification (or Receiver) algorithms, whose inputs and outputs are defined as follows. Upon input sk sampled from $D_{sk}(\eta, \mathscr{C})$ and $inp \in \mathbf{I}$, $S_{\mathscr{C}}(sk, inp)$ produces a set $(\mathcal{X}, m_{sk}, proof_{sk}, info_{sk})$, where $\mathcal{X}$ is a quantum register containing a state $|\psi_{sk}\rangle$ and the rest are classical messages. As mentioned previously, ($info_{sk}$ contains any additional information produced during the state generation process which is not needed for verifying the state and message). On input $(\mathcal{X}, m, proof)$ of correct size (otherwise the output is always 0), $R_{\mathscr{C}}(\mathcal{X}, m, proof)$ produces $b \in \{0,1\}$. By convention, the value for correctly prepared states is $b = 1$.

An outer protocol may rely on parts of the proof for future computations. In order to consider the most general application, the Receiver should therefore be authorised to ask adaptively for leaks of parts

[3] A qudit is a generalisation of a qubit where the state can be in superposition over d basis state, instead of $|0\rangle$ and $|1\rangle$ for a qubit.

[4] We call any protocol using Quantum Cut-and-Choose as a subroutine an outer protocol.

of the proof. This explicit treatment of leaks allows us to finely characterise which parts of the proof can be revealed without breaking the Sender's security in the Q-CC Protocol. Let $N = \#proof_{sk}$ be the number of formal messages contained in (an honestly generated) $proof_{sk}$ and let $\mathcal{L}_{\mathscr{C}} \subset \wp(N)$ be the tolerated leakage set (where $\wp(N)$ is the power set of $[N]$). The Receiver is allowed to ask for $leak \in \mathcal{L}_{\mathscr{C}}$, at which point the Sender chooses an element $l \in leak$ and sends back $proof_{sk,l}$, which is the l^{th} formal message contained in $proof_{sk}$. This choice is dependent (in the honest case) on the outer protocol that uses Q-CC as a subroutine and is modelled as a polynomial-time classical deterministic algorithm $L_{\mathscr{C}}$ which, given *leak*, *sk*, *inp*, *m*, *proof* and *info*, returns the aforementioned index l (which may also be equal to $\perp$ in case of an invalid leak request). Essentially, if there is no $leak \in \mathcal{L}_{\mathscr{C}}$ and $l, l' \in leak$ such that $proof_{sk,l}$ and $proof_{sk,l'}$ are leaked in the same execution, then the protocol is guaranteed to remains secure. This leakage model is fully general as more complex behaviour can be defined by incorporating it in the leakage choice function $L_{\mathscr{C}}$.

Intuition about States, Messages and Proofs. The presentation is purposefully kept very general so as to encompass a wide variety of situations. As an example, in the context of a state and message in the outer protocol, the state may be used in a later round for computations with the message containing instructions, encrypted or not, specifying which operations should be applied to it. Intuitively the proof reveals a lot of information about the state since it allows the Receiver to verify that it is correct (and it may be impossible to do so without it, since otherwise the Q-CC Protocol is not needed if the Receiver can test this in another way), but the Sender may not want to give this information to the Receiver since revealing it may compromise its security once this state is used later.

If m_{sk} contains in fact committed instructions, the Sender can use them to drive the computation by opening the corresponding opening information in $proof_{sk}$ based on the transcript that it receives from the Receiver (this is one example of permitted leakage). This allows for interactivity based on the state and the message in a larger protocol as opposed to other approaches which do not require this interaction (see for example [88] in the case of Yao's Protocol).

4.2.2 Constraints on the Sender and Receiver CP-maps

The various elements presented above must satisfy four properties presented below in order to be admissible in the Quantum Cut-and-Choose protocol. The first two are required for the protocol to be correct, while the last two deal with the security against Malicious players. The CP-maps are then called Cut-and-Choosable (or CC-able).

Definition 4.1 (Cut-and-Choosable CP-maps)**.** *We say that CP-maps $S_{\mathscr{C}}$ and $R_{\mathscr{C}}$ are* Cut-and-Choosable *if, for all $sk \leftarrow D_{sk}(\eta, \mathscr{C})$ and $inp \in \mathbf{I}$, the sets produced are:*

- *Extractable (Definition 4.2);*
- *Leak-verifiable (Definition 4.3);*
- *Proof-collapsing (Definition 4.4);*
- *Key-indistinguishable (Definition 4.5, either computational or statistical).*

First of all, it must be possible to extract efficiently the classical description of the state from the message and proof. The extraction algorithm $E_{\mathscr{C}}$ verifies deterministically the message using the proof

and, if the check passes (i.e. the output of this algorithm is not $\perp$), outputs the classical description of the state that was contained in (m, proof). For correctness to hold, if the state and message are honestly generated then this must correspond to the classical description of the corresponding honestly generated state.

Definition 4.2 (Extractability)**.** *The sets produced by the Sender CP-map are said to be* extractable *if there exists a deterministic algorithm $E_{\mathscr{C}}$ that, upon inputs (m, proof), outputs $\psi \in \{0,1\}^* \cup \{\perp\}$ such that $E_{\mathscr{C}}(m_{sk}, \mathit{proof}_{sk}) = \psi_{sk}$ for honestly generated $(\mathcal{X}, m_{sk}, \mathit{proof}_{sk})$, i.e. with $\mathcal{X}$ containing the state $|\psi_{sk}\rangle$.*

We suppose without loss of generality that all classical checks are performed by this extraction algorithm (it outputs $\perp$ if these checks fail, regardless of whether a classical description of a state can be extracted). This then allows us to fix the behaviour of $R_{\mathscr{C}}$, given $(\mathcal{X}, m, \mathit{proof})$, as follows:

- It uses $E_{\mathscr{C}}$ and obtains ψ. If $\psi = \perp$ then $R_{\mathscr{C}}$ outputs $b = 0$ and stops.
- $R_{\mathscr{C}}$ applies a projective measurement defined by $\{\mathsf{I}_{\mathcal{X}} - |\psi\rangle\langle\psi|, |\psi\rangle\langle\psi|\}$ on register $\mathcal{X}$ and returns the output of the measurement (the first outcome being associated to bit-value 0 and the second to 1).

A consequence of Definition 4.2 is that for all quantum polynomial-time Senders $S^*_{\mathscr{C}}(sk, \mathit{inp}, \mathcal{Z}_{aux})$ generating $(\rho^*, m^*, \mathit{proof}^*)$, we have $\Pr[R_{\mathscr{C}}(\rho^*, m^*, \mathit{proof}^*) = 1] = 1 - \epsilon$ if and only if there exists a state classically described by ψ such that $\Pr[R_{\mathscr{C}}(|\psi\rangle, m^*, \mathit{proof}^*) = 1] = 1$ and ρ^* is at least ϵ-close to $|\psi\rangle$. Such pairs (m^*, proof^*) are called *classically-accepting* since they pass the deterministic checks from $E_{\mathscr{C}}$. This in turn implies that for all sets $(\mathcal{X}, m_{sk}, \mathit{proof}_{sk})$ honestly generated via $S_{\mathscr{C}}$, we always have $R_{\mathscr{C}}(\mathcal{X}, m_{sk}, \mathit{proof}_{sk}) = 1$.

The next definition enforces that the Receiver is capable of verifying on its own that any leak that it receives from the Sender actually corresponds to the message and state that they have obtained previously for a given index. If it is not satisfied, then the leakage is useless since a Malicious Sender could send anything instead of the requested leak.

Definition 4.3 (Verifiable Leakage)**.** *The sets produced by the Sender CP-map are said to have* verifiable leakage *if there exists a deterministic, polynomial-time classical deterministic algorithm $V_{\mathscr{C}}$ which, upon input (m, p, i), for all $i \in [N]$ such that there exists $\mathit{leak} \in \mathcal{L}_{\mathscr{C}}$ with $i \in \mathit{leak}$, returns 1 if and only if there exists proof such that (m, proof) is classically-accepting and $\mathit{proof}_i = p$.*

In order for the proof of security against Malicious Sender to go through, the proof for all correctly generated messages and states must be close to unique. This is defined in the quantum case through a property called collapsing, as formalised in the next definition using the concept of collapsing relations (Definition 3.3). Essentially, it captures the fact that a computationally-bounded Adversary is not able to distinguish whether a quantum register containing the proof for a known message has been measured in the computational basis of not (meaning that the state of this register was close to one which had already been measured).

Definition 4.4 (Collapsing Proofs)**.** *Let $E_{\mathscr{C}}$ be the extractor algorithm as defined above, and $\mathfrak{M}, \mathfrak{P}$ be the sets from which the honest messages and proofs are respectively drawn. The relation R is the subset*

of classically-accepting messages and proofs, defined by $(m, proof) \in R \Leftrightarrow E_{\mathscr{C}}(m, proof) \neq \perp$. *Then the CP-map* $S_{\mathscr{C}}$ *is said to have* ϵ_c*-collapsing proofs if relation* R *is* ϵ_c *proof-collapsing given* m.

Finally, no Distinguisher should not be able to determine whether a given state $|\psi\rangle$ and message m were generated using the same secret key as other messages and states. This should hold even when it has access to the proofs for those other messages and states, some limited leakage about the proof associated with $(|\psi\rangle, m)$ and the additional information *info*. We define this formally in the statistically-secure regime in Definition 4.5, the computational variant is obtained by restricting the Distinguisher to being a QPT machine.

Definition 4.5 (Statistical Key-Indistinguishability)**.** *We say that the CP-maps are statistically key-indistinguishable if, for any Distinguisher* $D^*_{\mathscr{C}}$*, the wining advantage of the Distinguisher* $\epsilon_k(\eta)$ *in the following game is negligible in* η*:*

1. *The Distinguisher chooses* $inp_0 \in \mathbf{I}$ *and sends it to the Challenger.*
2. *The Challenger samples* $sk_0 \leftarrow D_{sk}(\eta, \mathscr{C})$ *and* $sk_1 \leftarrow D_{sk}(\eta, \mathscr{C})$*, along with* $inp_1 \in_R \mathbf{I}$ *chosen uniformly at random.*
3. *The Challenger samples a bit* $b \in_R \{0,1\}$ *uniformly at random and produce honestly two sets:* $(\mathcal{X}^0, m^0_{sk_b}, proof^0_{sk_b}, info^0_{sk_b})$ *using* (sk_b, inp_b) *and* $(\mathcal{X}^1, m^1_{sk_1}, proof^1_{sk_1}, info^1_{sk_1})$ *using* (sk_1, inp_1).
4. *The Challenger sends* $(info^0_{sk_b}, \mathcal{X}^0, m^0_{sk_b})$ *and* $(\mathcal{X}^1, m^1_{sk_1}, proof^1_{sk_1})$ *to the Distinguisher.*
5. *The Distinguisher sends* $\{leak_i\}_i$ *to the Challenger (with* $leak_i \in \mathcal{L}_{\mathscr{C}}$ *and* $leak_i \neq leak_j$ *for all* $i \neq j$*). For all* i*, let* $l_i := L_{\mathscr{C}}(leak_i, sk_b, inp_b, m^1_{sk_b}, proof^1_{sk_b}, info^1_{sk_b})$*. If* $l_i \neq \perp$*, the Challenger sends* $proof^0_{sk_b, l_i}$ *to the Distinguisher.*
6. *The Distinguisher outputs a bit* b' *and wins if* $b' = b$

Comments on Key-Indistinguishability There are protocols which do not satisfy the property of key-indistinguishability and yet may nevertheless be used in the Q-CC Protocol, but for which the security and composability must to be proven independently (typically if a secret key is used in the creation of all messages and states but is also reused later without being revealed, such as the signing key of a signature scheme).

When combined with collapsing-proofs, Definition 4.5 automatically imposes that two activations of the CP-map $S_{\mathscr{C}}(sk, inp)$ do not produce the same state-message pair, up to negligible probability.[5]

If using a trusted setup such as a Common Reference String for $\mathscr{C}$, key-indistinguishability can be defined using a trapdoor to generate the second set in the case $b = 1$. The presentation above is chosen so as to keep the presentation in the Plain Model with no setup assumptions.

Note as well that we do not impose on the messages to not contain any information about the state, only that giving additional opened messages does not procure more information to the Distinguisher.

4.2.3 The Quantum Cut-and-Choose Ideal Functionality and Protocol

The Send-Blind-Correct-State f_{send} Ideal Functionality is presented in Ideal Functionality 16. It certifies to the Receiver that a quantum register $\mathcal{X}$ and a message m were prepared and sent correctly without

[5] There are no guarantees about the adversarial advantage in this game if the Adversary has access to the full proof for the first set, which, due to collapsing, would be the case if the same first message is produced twice.

revealing the proof (for example, the classical description of the state) that would allow it to check it on its own.

Ideal Functionality 16 Send Blind Correct State

Inputs: The honest Sender sends $inp \in \mathbf{I}$ and the Receiver sends a dummy input λ to the Ideal Functionality. A corrupted Sender may send any $(m, proof)$ of its choice.
Computation by the Functionality:

- If the Ideal Functionality receives $inp \in \mathbf{I}$ from the Sender:
  1. It samples $sk \leftarrow D_{sk}(\eta, \mathscr{C})$ and runs $S_{\mathscr{C}}(sk, inp)$, obtaining $(\mathcal{X}, m_{sk}, proof_{sk}, info_{sk})$.
  2. It sends $(\mathcal{X}, m_{sk})$ to the Receiver and $(m_{sk}, info_{sk})$ to the Sender.
- If the Ideal Functionality receives $(m, proof)$ from the Sender:
  1. The Ideal Functionality computes $\psi = E_{\mathscr{C}}(m, proof)$. If $\psi = \perp$ (i.e. if the set $(m, proof)$ is not classically accepting), it sends Corrupted to the Sender and Receiver.
  2. Otherwise it initialises a quantum register $\mathcal{X}$ with $|\psi\rangle$ and sends $(\mathcal{X}, m)$ to the Receiver.
- If it has not sent Abort or Corrupted and $\mathcal{L}_{\mathscr{C}} \neq \emptyset$ then it initialises *leak* to the empty sequence (representing the set of leak requests).

Each leakage request is treated as separate Ideal Functionality call which inherits an internal state from the previous one (using sequential composability). It allows the Receiver to ask (potentially adaptively) for additional information (parts of the proof) about the state and message if some leakage is allowed.

Ideal Functionality 17 Proof Leakage

Inputs: The Receiver sends leak request $req \in \mathcal{L}_{\mathscr{C}}$.
Leakage by the Functionality:

- If $req \in \mathcal{L}_{\mathscr{C}}$ with $req \notin leak$, it sends req to the Sender (otherwise it sends Abort).
- If the Sender is honest it sends a dummy input and the Ideal Functionality computes the value $l := L_{\mathscr{C}}(req, sk, inp, m_{sk}, proof_{sk}, info_{sk})$. It updates *leak* to $leak \cup \{req\}$ and sends $(proof_l, l)$ to the Sender and the Receiver. If $l = \perp$, it sends $\perp$ to both parties.
- If the Sender is Malicious, it sends $l \in req$ to the Ideal Functionality, who sends $(proof_l, l)$ to the Receiver. The Sender can also choose to send Corrupted, which is then forwarded to the Receiver.

Note that the Ideal Functionality does not return the full $proof_{sk}$ to the honest Sender as doing so would prevent the construction of a Simulator later on (as it would give too much information to the Distinguisher and break Definition 4.5). This is also the reason for treating the leaks as part of the Ideal Functionality: if it were to give $proof_{sk}$ to the honest Sender then there would be no point for it to handle the leaks thanks to the verifiable leak property of CC-able maps (Definition 4.3).

Protocol 7 corresponds to the Q-CC Protocol realising Ideal Functionalities 16 and 17. Its correctness follows directly from the Extractability property of CC-able CP-maps (Definition 4.2). We study the security of the protocol in the next Section.

4.2.4 Security of the Quantum Cut-and-Choose Protocol

Theorem 4.3 states the security properties of Protocol 7. We first prove the security against a Malicious Receiver with negligible bounds and then the security against a Malicious Sender with an inverse-

Protocol 7 Quantum Cut-and-Choose

Inputs: The Sender has input $inp \in \mathbf{I}$. The Receiver has no input.

Protocol:

1. The Sender samples $sk \leftarrow D_{sk}(\eta, \mathscr{C})$.
2. The Sender then runs s times $S_{\mathscr{C}}(sk, inp)$ and obtains $\left\{\mathcal{X}^i, m^i_{sk}, proof^i_{sk}, info^i_{sk}\right\}_{i\in[s]}$.
3. The Sender sends all s labelled quantum registers $\left\{\mathcal{X}^i\right\}_{i\in[s]}$ and messages $\left\{m^i_{sk}\right\}_{i\in[s]}$ to the Receiver.
4. The Receiver chooses uniformly at random an index $\alpha \in_R [s]$ and sends it to the Sender.
5. The Sender sends $proof^i_{sk}$ for all $i \neq \alpha$ to the Receiver.
6. The Receiver checks that all received triplets $(\mathcal{X}^i, m^i_{sk}, proof^i_{sk})$ are of the correct format, otherwise it outputs Abort. It then computes $b^i = R_{\mathscr{C}}(\mathcal{X}^i, m^i_{sk}, proof^i_{sk})$ for all $i \neq \alpha$.
7. If there exists $i \neq \alpha$ such that $b^i = 0$, the Receiver sets its output to Corrupted and sends it to the Sender.
8. If $\mathcal{L}_{\mathscr{C}} \neq \emptyset$, the Sender initialises *leak*, representing a (for now empty) set of leak requests.

Leaks:

1. The Receiver sends leak request $req \in \mathcal{L}_{\mathscr{C}}$ with $req \notin leak$ to the Sender.
2. The Sender computes $l := L_{\mathscr{C}}(req, sk, inp, m_{sk}, proof_{sk}, info_{sk})$. It updates $leak \leftarrow leak \cup \{req\}$ and sends $(proof_l, l)$ to the Receiver. If $l = \perp$, it sends $\perp$ to the Receiver.
3. The Receiver computes $b^R_l := V_{\mathscr{C}}(m^\alpha, proof_l, l)$ and outputs Corrupted if $b^R_l = 0$.

Outputs: If neither party has set their output to Corrupted or Abort, the Sender's output is the set $\left(m^\alpha_{sk}, info^\alpha_{sk}, \left(proof^\alpha_{leak_i}\right)_i, leak\right)$ while the Receiver's output is $\left(\mathcal{X}^\alpha, m^\alpha_{sk}, \left(proof^\alpha_{leak_i}\right)_i, leak\right)$.

polynomial bound. Technical details regarding the quantum rewinding techniques used in these proofs are deferred to the next subsection.

Theorem 4.3. *Let $S_{\mathscr{C}}$ and $R_{\mathscr{C}}$ be CC-able CP-maps (Definition 4.1) with negligible key-distinguishing advantage ϵ_k (Definition 4.5) and negligible the collapsing-proof advantage ϵ_c (Definition 4.4). The Quantum Cut-and-Choose Protocol 7 realises Send-Blind-Correct-State and Proof Leakage Ideal Functionalities $\mathcal{O}(s\epsilon_k)$-securely against a Malicious Receiver (computational or statistical depending on the variant of key-indistinguishability) and $1/\sqrt{s} + \mathcal{O}(s\epsilon_c)$-securely against a computationally-bounded Malicious Sender.*

Proof of Security against Malicious Receiver The Simulator against an adversarial Receiver has single-query access to an oracle $O^{f_{send}}$ which implements the Send-Blind-Correct-State f_{send} Ideal Functionality and works as described in Simulator 1 below. If the Adversary sends Abort at any step, it is forwarded to the Ideal Functionality. After stopping, the Simulator outputs whatever the Adversary's output is.

We start by using the key-indistinguishability of CC-able CP-maps (Definition 4.5) to transform the interaction during simulation using a series of hybrid arguments into one where all the sets are generated using the same key chosen by the Ideal Functionality. The distinguishing advantage between this new simulation and the previous one is $(s-1)\epsilon_k$.

We can now analyse the effect of rewinding on the Adversary's state in this transformed interaction. It should be noted that *no information is kept between rewinds* as it is only used to ensure that the Adversary is forced to pick the set given to the Simulator by the Ideal Functionality as the evaluation set (as opposed to a checked set). The success probability of the step in the simulation that requires

Simulator 1 Q-CC Malicious Receiver

1. The Simulator calls the Ideal Functionality with dummy input λ. It receives as a result $(\mathcal{X}, m_{sk})$ (with $\mathcal{X}$ being a quantum register containing $|\psi_{sk}\rangle$), generated by the Ideal Functionality using the honest Sender CP-map $S_{\mathscr{C}}$ for a secret key sk chosen by the Ideal Functionality and input inp chosen by the honest Sender.
2. The Simulator sets $(\mathcal{X}^1, m^1) := (\mathcal{X}, m_{sk})$.
3. The Simulator samples a secret key $sk' \longleftarrow D_{sk}(n, \mathscr{C})$ and inp' uniformly at random from $\mathbf{I}$. It then runs $S_{\mathscr{C}}$ $s-1$ times to get $\left\{\mathcal{X}^i, m^i_{sk'}, proof^i_{sk'}\right\}_{i\in\{2,\ldots,s\}}$.
4. It chooses a random permutation π over $[s]$ and applies it to the classical communication registers of the Adversary and to $\left\{\mathcal{X}^i\right\}_{i\in[s]}$. Let $\alpha = \pi(1)$. The registers $\left\{\mathcal{X}^i\right\}_{i\in[s]}$ are then treated as internal registers of the Adversary (it has received the state). After that it applies the next unitary corresponding to the activation of the Adversary $R^*_{\left\{m^i_{sk_i}\right\}_{i\in[s]}}$ as in any other round.
5. It measures the classical communication register $\mathcal{C}$ of the Adversary using a projective measurement defined by $\{|\alpha\rangle\langle\alpha|, \mathsf{I}_{\mathcal{C}} - |\alpha\rangle\langle\alpha|\}$.
6. If the result of the previous measurement is 1 (the Adversary has not chosen α as the index of the unchecked state-message set) it rewinds the simulation to before step 4 (it applies the inverse of the Adversary's activation unitary and the inverse of the permutation), picks a different permutation and repeats the previous steps.
7. Otherwise (if the Adversary has chosen set α) the Simulator sends $proof^i_{sk'}$ for $i \neq \alpha$ to the Adversary. These correspond to correctly prepared sets and thus all checks pass.
8. During the second part of the execution, it transmits whatever leak the Adversary requests to the Ideal Functionality and relays any message from the Ideal Functionality back to the Adversary, until it either receives Corrupted from the Ideal Functionality or End from the Adversary, at which point it returns whatever the Adversary outputs and stops.

rewinding, as proven in Lemma 4.2, is $p = 1/s$, which is constant and independent from the internal state of the Adversary. The oblivious quantum rewind technique from [130] is therefore applicable for this step of simulation. After the rewinding procedure (once once the Adversary has chosen set α), we have that $\langle\phi_0(\psi)|\rho(\psi)|\phi_0(\psi)\rangle \geq 1-\epsilon_{rew}$ where $|\psi\rangle$ is the state of the Adversary before this step, $|\phi_0(\psi)\rangle$ corresponds to the state after a successful simulation happening in one try while $\rho(\psi)$ is the state at the end of the rewinding process.

The expected number of rewinds is $\mathcal{O}\left(\frac{\log(1/\epsilon_{rew})}{p(1-p)}\right)$, which here gives $\mathcal{O}\left(\frac{s^2}{s-1}\log(1/\epsilon_{rew})\right)$ for $p = 1/s$. Here ϵ_{rew} should be negligible in η, so $log(1/\epsilon_{rew}) = \mathcal{O}(\eta)$ is sufficient and $\mathcal{O}(\eta s)$ rewinds are required. More details may be found in Section 4.2.5.1.

Combining the distinguishing advantage due to key-indistinguishability and the one due to rewinding, we get that the total distinguishing advantage of the Environment is at most $(s-1)\epsilon_k + \epsilon_{rew}$. Setting the rewinding cost to $\epsilon_{rew} = \mathcal{O}(s\epsilon_k)$ yields the desired result.

■

Proof of Security against Malicious Sender Recall that the Adversary has quantum registers for its internal state and classical and quantum messages and that measurements are performed in the computational basis for obtaining classical values. Without loss of generality, we separate the classical messages and denote $\mathcal{M}^i$ the registers containing the CC-able messages and $\mathcal{P}^i$ the registers containing the proofs (then $\mathcal{M} := \bigotimes_i \mathcal{M}^i$ and similarly for $\mathcal{P}$). The Simulator against an adversarial Sender with single-query oracle access to the Send-Blind Correct State Ideal Functionality is defined in Simulator 2.

If the Adversary sends Abort at any step, it is forwarded to the Ideal Functionality. After stopping, the Simulator outputs whatever the Adversary's output is.

Simulator 2 Q-CC Malicious Sender

1. It runs the first step of the Adversary S^* and receives quantum registers $\{\mathcal{X}^i\}_{i\in[s]}$ and messages $\{m^i\}_{i\in[s]}$ by measuring all $\mathcal{M}^i$.
2. It chooses an index $\alpha \in_R [s]$ at random. Then:
   a) It runs the second step of the purified Adversary S^*_α. By measuring the registers $\mathcal{P}^i$ for $i \neq \alpha$, it receives the values $proof^i$ for $i \neq \alpha$.
   b) It checks whether the sets $i \neq \alpha$ are correct using the Receiver's checking algorithm $R_{\mathscr{C}}$, each outputting b^i. If there exists i such that $b^i = 0$, it sets a flag **corr** to $\perp$.
3. It rewinds the simulation by running the inverse of the second step of the Adversary $(S^*_\alpha)^\dagger$.
4. It then repeats the process described in the second step with a second index $\alpha' \in_R [s] \setminus \alpha$:
   a) It runs $S^*_{\alpha'}$ and receives $proof'^i$ for $i \neq \alpha'$ by measuring $\mathcal{P}^i$ for $i \neq \alpha'$.
   b) If there exists an index $i \notin \{\alpha, \alpha'\}$ such that $proof^i \neq proof'^i$, the Simulator aborts.
   c) It checks whether the set α is correct using the $R_{\mathscr{C}}$. If not, it sets **corr** to $\perp$.
5. If **corr** $= \perp$, it sends Corrupted to the Ideal Functionality and the Adversary and stops. Otherwise, it sends to the Ideal Functionality $(m^{\alpha'}, proof^{\alpha'})$ (it knows the proof for set α' by measuring $\mathcal{P}^{\alpha'}$ after running S^*_α.
6. For each leak-request (until it receives End from the Ideal Functionality, at which point it forwards it to the Adversary and stops):
   a) It forwards to the Adversary the request *req* received from the Ideal Functionality.
   b) It receives from the Adversary $(\tilde{p}, l)$, checks that $\tilde{p} = proof_l^{\alpha'}$ and if so it sends l to the Ideal Functionality. Otherwise it sends Corrupted to the Ideal Functionality and the Adversary and stops.

Reduction of the Sigma-Protocol. The protocol has three rounds of communication: commitment (sending the messages and states), challenge (choosing the random α) and response (sending the corresponding proofs). Protocols of this form are called *sigma-protocols*. We analyse it in two stages, similar to those applied in [129]. In the first part of the analysis, we use the collapsing proof property to simplify the operations performed by the Simulator in a way that is computationally equivalent to any Distinguisher. The second step applies the result from Section 4.2.5.2 to this simplified Simulator to show that the operations performed up to and including step 4 are indistinguishable from the Q-CC Protocol.

Recall that $\mathcal{W}_\mathcal{A}$ is the Adversary's internal register and we assume that the Adversary has already sent the states and messages. Therefore its operations act as identity on registers $\mathcal{X}^i$ and the registers $\mathcal{M}^i$ can only classically control other operations after being measured. The Simulator can be written formally

as follows, with α and α' chosen as above:

$$
\begin{aligned}
&\forall i, m^i \leftarrow \mathbb{M}(\mathcal{M}^i);\\
&S^*_\alpha(\mathcal{W}_\mathcal{A}\mathcal{MP});\ \forall i \neq \alpha,\ \mathit{proof}^i \leftarrow \mathbb{M}(\mathcal{P}^i),\ \psi^i \leftarrow E_\mathscr{C}(m^i, \mathit{proof}^i),\ \mathbb{M}_{\psi^i}(\mathcal{X}^i);\ (S^*_\alpha)^\dagger(\mathcal{W}_\mathcal{A}\mathcal{MP});\\
&S^*_{\alpha'}(\mathcal{W}_\mathcal{A}\mathcal{MP});\ \forall i \neq \alpha',\ \mathit{proof}'^i \leftarrow \mathbb{M}(\mathcal{P}^i),\ \psi^\alpha \leftarrow E_\mathscr{C}(m^\alpha, \mathit{proof}'^\alpha),\ \mathbb{M}_{\psi^\alpha}(\mathcal{X}^\alpha);\\
&ok_S \leftarrow (\forall i \notin \{\alpha, \alpha'\},\ \mathit{proof}^i = \mathit{proof}'^i) \wedge (\forall i,\ b^i \neq 0)
\end{aligned}
\tag{4.1}
$$

The simulation succeeds if ok_S is true. We operate the following transformations on the operations above:[6]

1. We start by asserting that $\forall i \notin \{\alpha, \alpha'\}$, $\mathit{proof}^i = \mathit{proof}'^i$ is true conditioned on the fact that $\forall i, b^i \neq 0$. An Adversary that is able to provide two valid proofs for a given message would break the proof-collapsing property (since such an Adversary can also prepare the quantum register containing the proof with a superposition of these two proofs and trivially distinguish if it was measured or not). The probability of this event is therefore bounded by $(s-2)\epsilon_c$ and the acceptance condition can be simplified to only test $\forall i,\ b^i \neq 0$ at the same distinguishing cost.
2. After the measurement of the registers $\mathcal{X}^i$, we add a measurement $\mathbb{M}_R$ on registers $(\mathcal{X}^i, \mathcal{M}^i, \mathcal{P}^i)$ defined by $\{\mathcal{R}_\mathscr{C}, \mathsf{I} - \mathcal{R}_\mathscr{C}\}$ where $\mathcal{R}_\mathscr{C}$ is a projector onto the subspace of correctly prepared states $\{|\psi\rangle\langle\psi| \otimes |mp\rangle\langle mp| \mid E_\mathscr{C}(m,p) = \psi \neq \perp\}$. This does not affect the state since these registers have already been measured right before this new measurement and are already either in or orthogonal to the subspace above.
3. Seeing as the proof has been measured at this stage, the effect and result of measuring the registers $\mathcal{X}^i$ using $\mathbb{M}_{\psi^i}$ or measuring $(\mathcal{X}^i, \mathcal{M}^i, \mathcal{P}^i)$ using $\mathbb{M}_R$ are perfectly identical. It is therefore possible to remove the measurement $\mathbb{M}_{\psi^i}$ and redefine b^i as the output of measurement $\mathbb{M}_R$ on $(\mathcal{X}^i, \mathcal{M}^i, \mathcal{P}^i)$. We write $\mathbb{M}_{R,\alpha}$ to indicate that the same operation is applied on all registers $i \neq \mathit{alpha}$. This operation outputs a single bit $ok = \bigwedge_{i \neq \alpha} b^i$ equal to 1 if and only if all checks pass.
4. Notice now that the value of ψ_i is no longer used, meaning that we can remove the call to the algorithm $E_\mathscr{C}$.
5. Similarly to [129], $\mathbb{M}_R$ and $\mathbb{M}(\mathcal{P}^i)$ commute so we can reverse the order in which these operations are performed: first check that the register $\mathcal{P}^i$ contains classically-accepting proofs and that the states are correct and only then measure it in the computational basis. At this point, the operations presented above are computationally-indistinguishable from:

$$
\begin{aligned}
&\forall i, m^i \leftarrow \mathbb{M}(\mathcal{M}^i);\\
&S^*_\alpha(\mathcal{W}_\mathcal{A}\mathcal{MP});\ ok \leftarrow \mathbb{M}_{R,\alpha}(\mathcal{XMP});\ \forall i \neq \alpha,\ \mathit{proof}^i \leftarrow \mathbb{M}(\mathcal{P}^i);\ (S^*_\alpha)^\dagger(\mathcal{W}_\mathcal{A}\mathcal{MP});\\
&S^*_{\alpha'}(\mathcal{W}_\mathcal{A}\mathcal{MP});\ ok' \leftarrow \mathbb{M}_{R,\alpha'}(\mathcal{XMP});\ \forall i \neq \alpha',\ \mathit{proof}'^i \leftarrow \mathbb{M}(\mathcal{P}^i);\\
&ok_S \leftarrow ok \wedge ok'
\end{aligned}
\tag{4.2}
$$

[6] Although similar to the ones presented in [129] in the proof of Theorem 43, our approach here is more general since we need to show that the states are indistinguishable. In [129] they are only concerned not to alter the winning probability of the Adversary and so some simplification do not apply in our case.

5. We can now use the collapsing-proof property to remove the measurement on registers $\mathcal{P}^i$, incurring a loss of $2(s-1)\epsilon_c$ in the process. This is true for both measurements since the verification maps $\mathbb{M}_{R,\alpha}$ and $\mathbb{M}_{R,\alpha'}$, if they succeed, are equivalent to restricting to valid Adversaries. Furthermore, the actions preceding both verifications by these maps can be incorporated in an Adversary's behaviour trying to distinguish whether the state has been measured. These therefore do not give more power to the Distinguisher since it could have applied these operations itself in the distinguishing process.
6. At the end, we can add an operation $(S^*_{\alpha'})^\dagger(\mathcal{W}_\mathcal{A}\mathcal{MP})$ since this can always be undone by a Distinguisher later. We then get:

$$
\begin{aligned}
&\forall i, m^i \leftarrow \mathbb{M}(\mathcal{M}^i);\\
&S^*_\alpha(\mathcal{W}_\mathcal{A}\mathcal{MP});\ ok \leftarrow \mathbb{M}_{R,\alpha}(\mathcal{XMP});\ (S^*_\alpha)^\dagger(\mathcal{W}_\mathcal{A}\mathcal{MP});\\
&S^*_{\alpha'}(\mathcal{W}_\mathcal{A}\mathcal{MP});\ ok' \leftarrow \mathbb{M}_{R,\alpha'}(\mathcal{XMP});\ (S^*_{\alpha'})^\dagger(\mathcal{W}_\mathcal{A}\mathcal{MP});\\
&ok_S \leftarrow ok \wedge ok'
\end{aligned}
\tag{4.3}
$$

We define the projector $P_\alpha := (S^*_\alpha)^\dagger \mathcal{R}_\mathscr{C}{}^\alpha S^*_\alpha$ and the associated projective measurement $\mathbb{M}_{P,\alpha} = \{P_\alpha, \mathsf{I} - P_\alpha\}$, the final operation can be written as:

$$
\forall i, m^i \leftarrow \mathbb{M}(\mathcal{M}^i);\ ok \leftarrow \mathbb{M}_{P,\alpha}(\mathcal{W}_\mathcal{A}\mathcal{XMP});\ ok' \leftarrow \mathbb{M}_{P,\alpha}(\mathcal{W}_\mathcal{A}\mathcal{XMP});\ ok_S \leftarrow ok \wedge ok' \tag{4.4}
$$

We then apply the result of Lemma 4.3 which shows that, under the conditions of special soundness (Definition 4.6, two transcripts are sufficient for the simulation) and computationally-unique responses (Definition 4.7, equivalent here to proof-collapsing), the distance between the real protocol and the simulation after the rewinding step is bounded by $1/\sqrt{s}$, evading issues naive rewinding faces due to no-cloning. In [127, 129] they only study the link between the probability that one measurement is successful and the probability that two consecutive measurements are successful. In comparison, we extend this result to show that if the second measurement is successful, then the resulting state is close to the one obtained after only one measurement.

Leak Simulation. Finally, we argue that the behaviour of the Simulator during the leaks is indistinguishable from the real case where the received leaks are verified using algorithm $V_\mathscr{C}$ since at this stage the Simulator has already measured the proof for index α' without aborting. This means that this proof is classically-accepting and therefore, if $\tilde{p} \neq \mathit{proof}_l^{\alpha'}$ with $\tilde{p}$ valid for algorithm $V_\mathscr{C}$, then the Sender would have found a collision for the message-proof relation. This would then also break the proof-collapsing requirement of CC-able maps, which can happen with probability at most ϵ_c.

Total Distinguishing Advantage. By combining the reductions, rewinding and leaks, after tracing out all but system α, the state at the end of the simulation is therefore $1/\sqrt{s} + \mathcal{O}(s\epsilon_c)$-close to the state at the end of the real protocol (where only $s-1$ states are checked and the remaining one is not measured).

■

Remark that the failure probability against an adversarial Sender goes from $1/s$ classically to $1/\sqrt{s}+\epsilon$ in the quantum case. In the context of Σ-Protocols, [26] closes this gap that was left open in [127]. But since they are mainly concerned with relating the success probability of the Adversary in two different games this is not directly applicable here: in the same way that we extend here the result of [127] to the distance between quantum states, the result of [26] – which is a direct improvement of [127] – would have to be similarly adapted. We leave the possibility of transposing their techniques as an open question.

These proofs provide examples where proving security against a quantum Adversary is not a straightforward translation of classical proofs, even for a fully classical functionality. It is not sufficient to use cryptographic primitives resistant against quantum computers (e.g. based on LWE), but proof techniques (and security parameters) must also be adapted. The part of the protocol that needs rewinding may even be entirely classical if the state is classical and thus the same proof (and extra cost in the case of the security against Malicious Sender) is necessary even for a fully classical CC protocol against quantum Adversaries (for a single evaluation set such as in [11]).

4.2.5 Analysis of Quantum Rewinding

Both proofs presented in the previous section use a technique called rewinding. Classically the Simulator runs the Adversary internally and rewinds it by having black box access to the *next message function* $m_{i+1} = \texttt{NM}(aux, m_1, \ldots, m_i)$ where aux is the Adversary's input and $(m_1, \ldots, m_i)$ are previous messages. It can then rewind the Adversary by saving all messages and sending them again, possibly changing the last one, to obtain different the results for different branches of the Adversary's program. This is impossible in general in the quantum setting due to no-cloning (quantum messages cannot be reused multiple times, nor the Adversary's state copied to be run again). However, two techniques given in [130] and [127] achieve a similar result, albeit with different additional constraints, and are applicable to the Simulators for the Q-CC Protocol.

4.2.5.1 Watrous' Oblivious Quantum Rewinding

We start by describing the setting required for using the oblivious quantum rewinding technique. Let Q be a unitary representing an attempt at simulating for some cheating Adversary whose internal state is in a pure state $|\psi\rangle$. Q is first applied to $|\psi\rangle\left|0^k\right\rangle$ and then the first qubit of the resulting state is measured in the computational basis to test whether the simulation succeeded or not. Let $p(\psi) \in (0,1)$ be the probability that this measurement outcome is 0, which corresponds to a successful simulation (result 1 indicates failure and requires a rewind). There are unique unit vectors $|\phi_0(\psi)\rangle$ and $|\phi_1(\psi)\rangle$ such that:

$$Q\,|\psi\rangle\left|0^k\right\rangle = \sqrt{p(\psi)}\,|0\rangle\,|\phi_0(\psi)\rangle + \sqrt{1-p(\psi)}\,|1\rangle\,|\phi_1(\psi)\rangle \tag{4.5}$$

Lemma 8 from [130] (restated here for completeness as Lemma 4.1) gives the conditions under which it is possible to construct a CP-map that uses Q as a subroutine and outputs a state arbitrarily close to $|\phi_0(\psi)\rangle$ for any initial state $|\psi\rangle$. It states that this is achievable without noticeably disturbing the Adversary's state if $p(\psi)$ is non-negligible and independent of ψ.

Lemma 4.1 (Quantum Rewinding Lemma, Exact Case, Taken from [130])**.** *Let Q be a (m,k)-quantum circuit and assume that $p = p(\psi)$ as defined above is constant over all choices of the input $|\psi\rangle$ and*

non-negligible. Then for every $\epsilon_{rew} > 0$ there is a general quantum circuit R such that for every input $|\psi\rangle$, the output $\rho(\psi)$ of circuit R satisfies $\langle\phi_0(\psi)|\rho(\psi)|\phi_0(\psi)\rangle \geq 1 - \epsilon_{rew}$, with:

$$\#R = \mathcal{O}\left(\frac{\log(1/\epsilon_{rew})\#Q}{p(1-p)}\right) \tag{4.6}$$

Rewinding therefore results in a state ϵ_{rew}-close to that of a successful simulation for any exponentially small ϵ_{rew} using $\mathcal{O}\left(\frac{\log(1/\epsilon_{rew})}{p(1-p)}\right)$ many rewinds, which is polynomial. Further details may be found in [130], proof of Lemma 8.

In the Q-CC Protocol, this rewinding technique is used when proving security against an adversarial Receiver. The associated simulation works as follows: the Simulator receives from the Ideal Functionality a state and message and sets them to index 1. It samples a secret key sk' and produces the sets $\left\{\mathcal{X}^i, m^i_{sk'}, proof^i_{sk'}\right\}_{i\in\{2,\ldots,s\}}$ using the honest Sender CP-map $S_{\mathscr{C}}$. It applies a permutation on the indices (let $\hat{\alpha} = \pi(1)$) and sends the sets to the Adversary. The adversarial Receiver chooses an index α and the Simulator is supposed to reveal all the proofs for all executions $i \neq \alpha$. If $\hat{\alpha} = \alpha$ all is good as the Simulator can reveal the honestly prepared proofs, which will pass the tests and the Adversary has no choice but to accept the set $\hat{\alpha}$. Otherwise the Simulator has to rewind and choose a different permutation. We apply the rewinding technique to a modified version where all state are generated using the same key (this modification is achieved using a hybrid argument and the key-indistinguishability property). The following Lemma 4.2 shows that the oblivious rewinding technique is applicable in this scenario.

Lemma 4.2 (Rewindable Receiver Simulation)**.** *The success probability of the step in Simulator 1 which requires rewinding is equal to $1/s$ independently of the internal state of the Adversary.*

Proof. After the adversarial Receiver sends the evaluation index α, before verifying if the simulation has succeeded or not, the state of the system is in product form (i.e. the random choices of $\alpha, \hat{\alpha}$, are depicted as an equal superposition, but are totally uncorrelated):

$$|\Phi_f\rangle := \left(\sum_{\alpha=1}^{s} c_\alpha |\phi(\psi,\alpha)\rangle |\alpha\rangle\right)\left(\sum_{\hat{\alpha}=1}^{s} \frac{1}{\sqrt{s}} |\hat{\alpha}\rangle\right) \tag{4.7}$$

where $\sum_\alpha |c_\alpha|^2 = 1$ are coefficients, $|\phi(\psi,\alpha)\rangle$ is the (normalised) state at the end of the protocol given initial state ψ and choice of graph α, while the last part ($|\hat{\alpha}\rangle$) corresponds to the random choice of set made by the Simulator. The projection to the subspace that does not need rewinding (where $\alpha = \hat{\alpha}$) is given by $P_0 := \sum_{\alpha'=1}^{s} \mathsf{I} \otimes |\alpha'\rangle\langle\alpha'| \otimes |\alpha'\rangle\langle\alpha'|$. We have:

$$|\Phi_f\rangle = P_0 |\Phi_f\rangle + (\mathsf{I} - P_0) |\Phi_f\rangle \tag{4.8}$$

We now need to bring it in the form of Eq.(4.5). We first define the (normalised) states:

$$\begin{aligned} |\phi_0(\psi)\rangle &= \sum_{\alpha=1}^{s} c_\alpha |\phi(\psi,\alpha)\rangle |\alpha\rangle |\alpha\rangle \\ |\phi_1(\psi)\rangle &= \sum_{\alpha=1}^{s}\sum_{\hat{\alpha}\neq\alpha} \frac{c_\alpha}{\sqrt{s-1}} |\phi(\psi,\alpha)\rangle |\alpha\rangle |\hat{\alpha}\rangle \end{aligned} \tag{4.9}$$

Then we have:

$$|\Phi_f\rangle = \sqrt{\frac{1}{s}}\,|\phi_0(\psi)\rangle + \sqrt{1-\frac{1}{s}}\,|\phi_1(\psi)\rangle \tag{4.10}$$

Now, following the unitary action that led to $|\Phi_f\rangle$, the measurement $\{P_0, \mathsf{I} - P_0\}$ is performed and the outcome is stored in the value of an extra qubit (the first one):

$$|\Phi_f\rangle = \sqrt{\frac{1}{s}}\,|0\rangle\,|\phi_0(\psi)\rangle + \sqrt{1-\frac{1}{s}}\,|1\rangle\,|\phi_1(\psi)\rangle \tag{4.11}$$

This is exactly in the form of Eq. (4.5) where $p(\psi) = 1/s$ is constant (independent of ψ).

■

4.2.5.2 Extending Unruh's Quantum Rewinding

Watrous' lemma only ensures that the simulation is successful, but *no information* is kept between two rewinds (hence *oblivious rewinding*). The Simulator against the Malicious Sender in the Cut-and-Choose protocol needs two transcripts in order to recover the proofs corresponding to the evaluation index (which are secret in a normal execution), so another type of rewinding is necessary. We describe here the rewinding result of [127, 129] and give a new characterisation in terms of trace distance between two states (instead of the original presentation focusing on the winning probability of an Adversary in a security game).

Let Π be a protocol between a Prover, with input (x, w), and a Verifier, with input x and producing an output bit, exchanging three messages (called sigma-protocol): commitment *com* by the Prover, challenge *ch* sampled (efficiently) uniformly at random by the Verifier from the set C_x (membership in C_x has to be easy to decide), and response *resp* by the Prover. The Verifier decides whether to accept or reject (outputting 1 or 0 respectively) using a deterministic polynomial-time computation on $(x, com, ch, resp)$ (if the output is 1 it is called an *accepting conversation* for x).

We suppose that such protocols satisfy the following two properties. Special soundness (Definition 4.6) means that given two correct communication transcripts with different challenges, an extractor is able to compute a witness (in the security proof the Simulator recovers the values of all proofs given two transcripts). Quantum Computationally Unique Response (Definition 4.7) is obtained by applying collapsing relations (Definition 3.3) to the verification procedure of the sigma-protocol (seen as a relation between commitment, challenge and response). Here, the Distinguisher cannot detect whether the response register has been measured if the commitment and challenge registers have been measured.

Definition 4.6 (Special Soundness)**.** *Such a protocol satisfies* special soundness *if there exists a deterministic polynomial-time algorithm* K_0 *(the* special extractor*) such that for any two accepting conversations* $(com, ch, resp)$ *and* $(com, ch', resp')$ *for* x *with* $ch \neq ch'$*,* $w := \mathsf{K}_0(x, com, ch, resp, ch', resp')$*.*

Definition 4.7 (Quantum Computationally Unique Response [38][7])**.** *Let* Σ *be a sigma-protocol as defined above, with* Verif_Σ *being the verification procedure performed by the Verifier, and* $\mathfrak{M}_{com}$*,* $\mathfrak{M}_{ch}$ *and* $\mathfrak{M}_{resp}$ *be the sets from which the commitment, challenge and response are drawn. Then* Σ *is said to*

[7] Equivalent to Definition 8 from [92].

have quantum computationally unique responses *if relation* Verif_Σ *is* ϵ_c^Σ *resp-collapsing given* (com, ch) *for all true statements* x *(meaning that there exists a witness* w *such that a Prover using* (x, w) *in the sigma-protocol makes the Verifier accept).*

We can now describe the Canonical Extractor which, using any Adversarial Prover that passes the verification test once, is able to extract a witness w by running it twice and using the special soundness property. Recall that each activation of the Adversary is a separate unitary operation and getting a message is modelled as a measurement in the computational basis.

Canonical Extractor [127]. The extractor runs the first step of the Adversary to recover *com*, chooses two values $ch, ch' \in C_x$, runs the second step with *ch* to get *resp*, applies the inverse of the second step (this is the rewinding) and reruns the second step with ch' to get $resp'$ before applying K_0. Each response is uniquely determined for a computationally-bounded Distinguisher after being collapsed by the commitment and the challenge. Therefore, if the measured response of the Adversary is correct then they must have sent a state close to the real response, hence the measurement does not disturb too much the internal state of the Adversary.

In the Q-CC Protocol, *com* corresponds to step 3 (Sender sends the states and messages), *ch* is step 4 (Receiver chooses and sends the evaluation index α) and *resp* is step 5 (revealing the proofs for indices $i \neq \alpha$). Given challenge α, let P_α – analogous to the P^*_{ch} in [127] – denote the projector corresponding to applying the adversary's unitary activation S^*_α, followed by projecting all the sets other than α in the correct subspace, and then applying $(S^*_\alpha)^\dagger$. Let P be the projector on the "correct" subspace that answer all s tests, i.e. $PP_\alpha := P$ for all α. Let $p_1 = \frac{1}{s}\sum_\alpha \mathrm{Tr}(P_\alpha \rho)$ and $p_2 = \mathrm{Tr}(P\rho)$. Consider the following two CPTP maps:

$$\begin{aligned} \Phi_1(\rho) &= \frac{1}{s}\sum_{\alpha=1}^{s} P_\alpha \rho P_\alpha + (1-p_1)\,|\mathsf{Abort}\rangle\langle\mathsf{Abort}| \\ \Phi_2(\rho) &= P\rho P + (1-p_2)\,|\mathsf{Abort}\rangle\langle\mathsf{Abort}| \end{aligned} \tag{4.12}$$

The real protocol corresponds to the Sender and Receiver acting on some shared state ρ by applying the CPTP map $\Phi_1(\cdot)$, i.e. measuring if the sets sent are classically-accepting by choosing randomly one set to be left unmeasured. In the ideal world, the Simulator applies $\Phi_2(\cdot)$ to ρ, i.e. it verifies that all sets are correct. The following Lemma 4.3 shows that the aforementioned CP-maps are close, which implies the protocol's security.

Lemma 4.3 (Post-Rewind State)**.** *Let* $|\psi\rangle$ *be a purification of shared state* ρ *(with purification register* $\mathcal{W}$*) and let* $\{P_\alpha\}_{\alpha\in[s]}$ *and* P *be as defined above, then:*

$$\Delta\left((P\otimes \mathsf{I}_\mathcal{W})\,|\psi\rangle\langle\psi|\,(P\otimes \mathsf{I}_\mathcal{W}), \frac{1}{s}\sum_\alpha (P_\alpha\otimes \mathsf{I}_\mathcal{W})\,|\psi\rangle\langle\psi|\,(P_\alpha\otimes \mathsf{I}_\mathcal{W})\right) \leq \sqrt{\frac{1}{s}} \tag{4.13}$$

Proof. Let $\sigma_1 = \frac{1}{s}\sum_\alpha (P_\alpha\otimes \mathsf{I}_\mathcal{W})\,|\psi\rangle\langle\psi|\,(P_\alpha\otimes \mathsf{I}_\mathcal{W})$ and $\sigma_2 = (P\otimes \mathsf{I}_\mathcal{W})\,|\psi\rangle\langle\psi|\,(P\otimes \mathsf{I}_\mathcal{W})$, and recall that $p_1 = \frac{1}{s}\sum_\alpha \mathrm{Tr}(P_\alpha\rho)$ and $p_2 = \mathrm{Tr}(P\rho) = \langle\psi|P|\psi\rangle$. We will use in this proof the distance and fidelity notions for sub-normalised states presented in Section 2.2.1, in particular Equation 2.15.

The Q-CC Protocol satisfies the Special Soundness and Quantum Computationally Unique Response properties defined above. Special soundness means that any two tests, when both successful, allow the Simulator to recover a "witness" (in this case the proof for the non tested state-message pair). This leads to $P_\alpha P_{\alpha'} = P$ for all $\alpha \neq \alpha'$. On the other hand, the proof-collapsing property of CC-able CP-maps implies that there is (almost) a unique classical response to each challenge (this is guaranteed by Definition 4.4 of CC-able CP-maps). In terms of operators, it is equivalent to saying that two projections P_α and $P_{\alpha'}$ acting on the same subsystem, are either identity or themselves which essentially means that they commute, i.e. $P_\alpha P_{\alpha'} = P_{\alpha'} P_\alpha$. We further define P^c_α through the relation $P_\alpha = P + P^c_\alpha$. From special soundness, we have that:

$$P^c_\alpha P^c_{\alpha'} = (P_\alpha - P)(P_{\alpha'} - P) = P_\alpha P_{\alpha'} - P_\alpha P - P P_{\alpha'} + P = \delta_{\alpha,\alpha'} P^c_\alpha \tag{4.14}$$

Since P^c_α, $P^c_{\alpha'}$ and P are orthogonal and the sum of the traces (probabilities) of the P^c_α-terms cannot exceed $1 - p_2$, we then have that:

$$\begin{aligned} p_1 = \operatorname{Tr}\sigma_1 = \operatorname{Tr}\left(\frac{1}{s}\sum_\alpha (P_\alpha \otimes \mathsf{I}_{\mathcal{W}})\,|\psi\rangle\langle\psi|\,(P_\alpha \otimes \mathsf{I}_{\mathcal{W}})\right) &= \frac{1}{s}\sum_\alpha \langle\psi|(P + P^c_\alpha)\otimes \mathsf{I}_{\mathcal{W}}|\psi\rangle \\ &= p_2 + \frac{1}{s}\sum_\alpha \langle\psi|P^c_\alpha \otimes \mathsf{I}_{\mathcal{W}}|\psi\rangle \leq p_2 + \frac{1}{s}(1-p_2) \end{aligned} \tag{4.15}$$

It is straightforward to see that:

$$F\left((P\otimes \mathsf{I}_{\mathcal{W}})\,|\psi\rangle\langle\psi|\,(P\otimes \mathsf{I}_{\mathcal{W}}), \frac{1}{s}\sum_\alpha (P_\alpha \otimes \mathsf{I}_{\mathcal{W}})\,|\psi\rangle\langle\psi|\,(P_\alpha \otimes \mathsf{I}_{\mathcal{W}})\right) = p_2 \tag{4.16}$$

We then obtain for the sub-normalised fidelity:

$$\tilde{F}(\sigma_1,\sigma_2) \geq p_2 + \sqrt{(1-p_2)\left(1 - p_2 - \frac{1}{s}(1-p_2)\right)} \geq p_2 + (1-p_2)\left(1 - \frac{1}{2s}\right) \geq 1 - \frac{1}{2s} \tag{4.17}$$

It follows that:

$$\tilde{\Delta}(\sigma_1,\sigma_2) \leq \sqrt{1 - \left(1 - \frac{1}{2s}\right)^2} \leq \sqrt{\frac{1}{s}} \tag{4.18}$$

We have $\Phi_1(\rho) = \sigma_1 + (1-p_1)\,|\mathsf{Abort}\rangle\langle\mathsf{Abort}|$ and $\Phi_2(\rho) = \sigma_2 + (1-p_2)\,|\mathsf{Abort}\rangle\langle\mathsf{Abort}|$ and using the above expressions we get:

$$\Delta(\Phi_1(\rho), \Phi_2(\rho)) \leq \sqrt{\frac{1}{s}} \tag{4.19}$$

This result is independent of $|\psi\rangle$ (and p_2) and thus completes the proof.

■

We have $\Phi_1(\cdot) \overset{1/\sqrt{s}}{\approx} \Phi_2(\cdot)$ where $\Phi_1(\cdot)$ corresponds to the real protocol's operation (project in one of P_α subspace, randomly chosen from the s possible challenges), while $\Phi_2(\cdot)$ is the CP-map corresponding to the projection in the P subspace (which is applied in the simulation by making a measurement $\{P, \mathsf{I} - P\}$). The simulated view is therefore $1/\sqrt{s}$-close to the real protocol.

4.3 Exponentially-Secure Fraction Classical Cut-and-Choose

We will in this section (in contrast with the rest of this Chapter) define formally and prove the security of the *exponentially-secure Fraction Classical Cut-and-Choose* (FC-CC). We will therefore have s sets in total, k *check sets* and $s - k$ *evaluation sets*. The purpose of the protocol is then to guarantee that a certain fraction of the messages transmitted to the Client has been prepared correctly. As such we define $t \leq s - k$ to be the lowest number of sets that need to be corrupted by an Adversary to succeed in breaking an outer protocol that relies on this version of Classical CC. We suppose that the Adversary is flagged as cheating as soon as one invalid classical message has been detected in the check phase. The security in this section will be proven in the Abstract Cryptography framework detailed in Section 3.3.2. Note that, although the honest players are purely classical and the channels they interact with as well, we impose no such restriction on adversaries which may therefore use quantum machines however they wish.

We first define the Send-Blind-Correct-Fraction-of-Messages Ideal Functionality and associated properties are first defined, followed by the Cut-and-Choose protocol and finally the proofs of security against malicious Receiver and malicious Sender. For each value of t, we derive the optimal number of check sets that should be used to secure the CC protocol. Interestingly, in the case where $t = (s - k)/2$ and the Adversary needs to corrupt at least half of the evaluation sets to succeed (as in the case where a majority vote is used later), we find that the optimal number of tests is $3s/5$ instead of the most often used $s/2$.

4.3.1 New Constraints, Ideal Resource and Protocol Presentation

All previous definitions of the moving pieces of CC remain the same as in the previous Section, with the only difference being that the machines $CS_{\mathscr{C}}$ and $CR_{\mathscr{C}}$ are now PPT instead of QPT and do not produce quantum states but deal only with messages and proofs, i.e. $CS_{\mathscr{C}}(sk, inp)$ produces $(m_{sk}, proof_{sk}, info_{sk})$ and $CR_{\mathscr{C}}(m, proof)$ produces $b \in \{0, 1\}$. We do not deal with leaks explicitly in this setting to avoid overloading the description of the functionality and protocol (more problems may arise from dealing with leaks across multiple sets). The acceptable algorithms are then those that satisfy the following Definition 4.8.

Definition 4.8 (Fraction Cut-and-Choosable Algorithms)**.** *We say that the algorithms $CS_{\mathscr{C}}$ and $CR_{\mathscr{C}}$ are* Fraction Cut-and-Choosable *(of F-CC-able) if, for all $sk \leftarrow D_{sk}(\eta, \mathscr{C})$ and $inp \in \mathbf{I}$:*

- *It has an Indistinguishable Dual-Mode setup (Definition 4.9);*
- *The sets are Trapdoor Extractable (Definition 4.10);*
- *The sets are Key-indistinguishable (Definition 4.5, either computational or statistical) even with the full proof for all sets.*

We first suppose that there exists two probabilistic polynomial-time algorithms Setup and $\widetilde{\mathbf{Setup}}$ for producing the public string $\mathscr{C}$, the outputs of which are indistinguishable for any computationally-bounded distinguisher. However, $\widetilde{\mathbf{Setup}}$ also produces an additional information called trapdoor. This is captured in Definition 4.9. We will assume that both players have access to a Common Reference String Ideal Resource 6 that produces the string $\mathscr{C}$, created using the algorithm Setup.

Definition 4.9 (Indistinguishable Dual-Mode Setup)**.** *We say that the scheme has a* dual-mode setup *if there exists a negligible function $\epsilon_{setup}(\eta)$ and two algorithms Setup and $\widetilde{\mathsf{Setup}}$ that take as input the security parameter η and CC parameter s and output respectively $\mathscr{C}$ and $(\mathscr{C}, \mathfrak{T})$ such that for any auxiliary state ρ_{aux} and efficient quantum Distinguisher $\mathcal{D}$ we have:*

$$\left|\Pr\left[b = 0 \mid b \leftarrow \mathcal{D}\left(\rho_{aux}, \mathcal{O}^{\mathsf{Setup}(\cdot)}\right)\right] - \Pr\left[b = 0 \mid b \leftarrow \mathcal{D}\left(\rho_{aux}, \mathcal{O}^{\widetilde{\mathsf{Setup}}(\cdot)}\right)\right]\right| \leq \epsilon_{setup}(n) \tag{4.20}$$

In the equation above, the Distinguisher only has classical access to an oracle that either implements either Setup or $\widetilde{\mathsf{Setup}}$, the latter of which does not output the trapdoor.

Then, given the trapdoor $\mathfrak{T}$, it must be possible to extract efficiently the proof from any honestly generated message using an efficient deterministic extraction algorithm $TE_{\mathscr{C}}$. Note that alternative ways of extracting the proof from the message may be employed, for example by using an extractable commitment scheme to commit to the proof with the commitment being stored in the message. We use the definition below for convenience in order to keep the presentation of the protocol self-contained.

Definition 4.10 (Trapdoor Extractability)**.** *The sets produced by Sender algorithm are said to be* trapdoor extractable *if there exists an efficient deterministic algorithm $TE_{\mathscr{C}}$ that, upon inputs $(m, \mathfrak{T})$, outputs proof $\in \{0,1\}^* \cup \{\perp\}$ such that $TE_{\mathscr{C}}(m_{sk}, \mathfrak{T}) = proof_{sk}$ for honestly generated $(m_{sk}, proof_{sk})$. Furthermore, if $TE_{\mathscr{C}}(m, \mathfrak{T}) \neq \perp$ then $CR_{\mathscr{C}}(m, TE_{\mathscr{C}}(m, \mathfrak{T})) = 1$ and conversely if $TE_{\mathscr{C}}(m, \mathfrak{T}) = \perp$ there is no p such that $R_{\mathscr{C}}(m, p) = 1$.*

The Send Blind Correct Fraction of Messages $f_{send-frac}$ Ideal Resource which the exponential Classical Cut-and-Choose Protocol will be emulating is presented in Ideal Resource 18. The intuition for this Resource is that it allows the Receiver to be certain that a certain fraction of received messages were prepared and sent correctly (as opposed to a single message in the previous Section). Note that the Resource does not reveal which messages it has in fact checked as valid. We denote f the number of messages forwarded by the Resource to the Receiver and α the maximum tolerated fraction of defective messages among the transmitted messages.

The following Protocol 8 constructs Ideal Functionality 16 up to a negligible ϵ. Compared to the protocol presented in the previous Section, the Receiver will here test $k = s - f = \kappa s$ sets out of the s sets which are sent by the Sender for constant $\kappa \in (0, 1)$. The optimal value of this parameter will be determined later in the security proof. This protocol uses a Two-Party Coin-Tossing Resource 8 and Common Random String Resource 6. The coin-toss samples uniformly at random a subset of $[s]$ of size k corresponding to the check sets.

Resource 18 Send Blind Correct Fraction of Messages

Public Information: Number of forwarded messages f, maximum fraction of invalid sets α.
Inputs: The honest Sender sends $inp \in \mathbf{I}$ while a corrupted Sender may send any $\left\{\{m^i\}_{i\in[f]}, H \subseteq [f], \{proof^j\}\right\}_{j\in H}$ of its choice for $\#H > (1-\alpha)f$ (this is filtered by a bit c set to 0 in the honest case). The Sender and Receiver have a filtered bit interface, o_S and o_R respectively, indicating whether they abort or not, set to 0 if honest.
Computation by the Resource:

- If either o_S or o_R are set to 1, the Ideal Functionality outputs Abort to both parties.
- If the Ideal Functionality receives $inp \in \mathbf{I}$ from the Sender:
  1. It samples $sk \leftarrow D_{sk}(\eta, \mathscr{C})$ and runs f times $CS_{\mathscr{C}}(sk, inp)$, obtaining the sets $\left\{m^i_{sk}, proof^i_{sk}, info^i_{sk}\right\}_{i\in[f]}$.
  2. It sends $\{m^i_{sk}\}_{i\in[f]}$ to the Receiver and $\{m^i_{sk}, proof^i_{sk}, info^i_{sk}\}_{i\in[f]}$ to the Sender.
- If the Ideal Functionality receives $\left\{\{m^i\}_{i\in[f]}, H \subseteq [f], \{proof^i\}_{i\in H}\right\}$ from the Sender:
  1. The trusted party computes $b^i = CR_{\mathscr{C}}(m^i, proof^i)$ for all $j \in H$. If there is an index i such that $b^i = 0$, it sends Corrupted to the Sender and Receiver.
  2. Otherwise it sends $\{m^i\}_{i\in[f]}$ to the Receiver.

Protocol 8 Fraction Classical Cut-and-Choose

Inputs: The Sender has input $inp \in \mathbf{I}$. The Receiver has no input.
Protocol:

1. The Sender and Receiver perform a call to the Common Random String Resource 6 implementing the Setup algorithm, receiving the string $\mathscr{C}$.
2. The Sender samples $sk \leftarrow D_{sk}(\eta, \mathscr{C})$.
3. The Sender then runs s times $CS_{\mathscr{C}}(sk, inp)$ and obtains $\left\{m^i_{sk}, proof^i_{sk}, info^i_{sk}\right\}_{i\in[s]}$.
4. The Sender sends all s labelled messages $\left\{m^i_{sk}\right\}_{i\in[s]}$ to the Receiver.
5. The Sender and Receiver perform a call to the Two-Party Coin-Tossing Resource 8, receiving the set $C \subset [s]$ of size k corresponding to the indices of the check sets.
6. The Sender sends $\left\{proof^i_{sk}\right\}_{i\in C}$ to the Receiver.
7. The Receiver checks that all received sets $\left\{m^i_{sk}, proof^i_{sk}\right\}_{i\in C}$ are of the correct format, otherwise it outputs Abort. It then computes $b^i = CR_{\mathscr{C}}(m^i_{sk}, proof^i_{sk})$ for all $i \in C$.
8. If there exists $i \in C$ such that $b^i = 0$, the Receiver sets its output to Corrupted and sends it to the Sender.

Outputs: If neither party has set their output to Corrupted or Abort, the Sender's output are the sets $\{m^i_{sk}, proof^i_{sk}, info^i_{sk}\}_{i\in C^c}$ while the Receiver's output is $\{m^i\}_{i\in C^c}$, with $C^c = [s] \setminus C$.

4.3.2 Security of the Fraction Classical Cut-and-Choose Protocol

While the protocol's correctness is guaranteed by the Trapdoor Extractability (Definition 4.10) of F-CC-able algorithms, we prove in this subsection the security of the protocol above by showing that it constructs the Send-Blind-Correct-Fraction-of-Messages Resource with negligible distinguishing advantage for both parties.

Theorem 4.4. *Let $CS_{\mathscr{C}}$ and $CR_{\mathscr{C}}$ be F-CC-able algorithms (Definition 4.8) with ϵ_k and ϵ_{setup} respectively the negligible key-distinguishing advantage (Definition 4.5) and the distinguishing advantage of the dual-mode setup (Definition 4.9). For s linear in the security parameter, the Fraction Classical Cut-and-Choose Protocol 8 constructs the Send Blind Correct Fraction of Messages Resource $\mathcal{O}(s\epsilon_k)$-securely against a Malicious Receiver (computational or statistical depending on the variant of key-indistinguishability) and $\epsilon_S + \epsilon_{setup}$-securely against a computationally-bounded Malicious Sender, for negligible ϵ_S.*

Being in the (r_{CT}, r_{CRS})-hybrid model, the simulator will replace these resources, interfacing with the Adversary in their stead (of course this needs to be done in a way that is indistinguishable from what the Resources would have sent).

Security against malicious Receiver The Simulator Sim_{R^*}, which has single-query access to the Send Blind Correct Fraction of Messages Resource, is described in the following Simulator 3.

Simulator 3 FC-CC Malicious Receiver

1. The Simulator uses algorithm $\mathsf{Setup}(\eta, s)$ to produce $\mathscr{C}$. It sends $\mathscr{C}$ to the Adversary on its interface that is linked to r_{CRS} in the real protocol.
2. The Simulator calls the Send Blind Correct Fraction of Messages Resource. It receives as a result messages $\{m^i\}_{i\in[f]}$, honestly generated by the Resource using the honest Sender algorithm $CS_{\mathscr{C}}$ for a secret key sk chosen by the Resource and input inp chosen by the honest Sender.
3. The Simulator chooses uniformly at random a set $C \subset [s]$ of size k corresponding to the indices of the check sets.
4. The Simulator samples a secret key $sk' \longleftarrow D_{sk}(n, \mathscr{C})$ and inp' uniformly at random from $\mathbf{I}$. It then runs k times the algorithm $CS_{\mathscr{C}}$ to get $\left\{m^i_{sk'}, proof^i_{sk'}\right\}_{i\in C}$.
5. It sends $\{\tilde{m}^i\}_{i\in[s]}$ to the Adversary, such that $\tilde{m}^i = m^i_{sk'}$ for $i \in C$ and $\tilde{m}^i = m^i$ for $i \notin C$ (the ones received from the Resource, in order).
6. It sends C to the Adversary on its interface that is linked to r_{CT} in the real protocol.
7. The Simulator sends $proof^i_{sk'}$ for $i \in C$ to the Adversary and stops. These correspond to correctly prepared sets and thus all checks pass.

The set C is chosen from the exact same distribution as the Coin-Tossing Resource and as such this step is perfectly indistinguishable. Using the Key-Indistinguishability of F-CC-able algorithms (Definition 4.5) via a series of $k = \kappa s$ hybrid arguments to transform the interaction during simulation into one where all the sets are generated using the same key chosen by the Ideal Functionality, we incur a distinguishing cost of $\kappa s\epsilon_k$.

■

Security against malicious Sender The Simulator Sim_{S^*}, which has single-query access to the Send Blind Correct Fraction of Messages Resource, is described in Simulator 4

Simulator 4 FC-CC Malicious Sender

1. The Simulator uses algorithm $\widetilde{\mathsf{Setup}}(\eta, s)$ to produce $(\mathscr{C}, \mathfrak{T})$. It sends $\mathscr{C}$ to the Adversary on its interface that is linked to r_{CRS} in the real protocol.
2. The Simulator receives messages $\{m^i\}_{i\in[s]}$ from the Adversary.
3. It samples uniformly at random a set $C \subset [s]$ of size k corresponding to the indices of the check sets and sends it to the Adversary on its interface that is linked to r_{CT} in the real protocol.
4. It receives in return $proof^i$ for $i \in C$. If any set is inconsistent with what an honest Receiver is supposed to receive, the Simulator sends Abort to the Resource and Adversary and stops.
5. Otherwise, it computes $b^i = CR_{\mathscr{C}}(m^i, proof^i)$ for $i \in C$. If there exist $i \in C$ such that $b^i = 0$ it sends incorrectly generated messages and proofs of its choosing to the Resource (which outputs Corrupted to the other party). It transmits Corrupted to the Adversary and stops.
6. If all checks pass, it produces $p^i := TE_{\mathscr{C}}(m^i, \mathfrak{T})$ for all $i \notin C$ using the trapdoor $\mathfrak{T}$. If there are too many failed extractions, i.e. $\#\{i \mid p^i = \perp\} \geq \alpha(s-k)$, the Simulator outputs Fail and stops.
7. Otherwise, it sets $H = \{i \mid p^i \neq \perp\}$ and sends $\{\{m^i\}_{i\in[s-k]}, H \subseteq [f], \{p^i\}_{i\in H}\}$ to the Send Blind Correct Fraction of Messages Resource and stops.

We denote $\Pr[\mathsf{Fail}] = \epsilon_S$ the probability that the Simulator outputs Fail in the simulation above. Lemma 4.4 below shows that this probability is negligible in the security parameter η so long as s is linear in η. Conditioned on the Simulator not failing, the only distinguishing opportunity for any Environment lies in the difference between $\widetilde{\mathsf{Setup}}(\eta, s)$ and $\mathsf{Setup}(\eta, s)$. Its advantage is then at most ϵ_{setup}, leading to a combined advantage of $\epsilon_S + \epsilon_{setup}$. Note that the functionality guarantees that a fraction of the remaining sets have been prepared correctly, i.e. there exists an associated correct proof for the messages sent. It does not certify that the Adversary actually has access to these proofs, which is why the Simulator can send the proofs it has extracted without knowing whether the Adversary has the same ones.[8]

■

Suppose that the Adversary needs to attack at least t messages to corrupt the overall computation. We will calculate in the following lemma the optimal number of checks that the honest Receiver needs to perform (that minimises the probability of successful cheating). As stated previously, there are s messages in total and k of them are checked by the Receiver.

Lemma 4.4 (Optimal Number of Checks)**.** *Suppose that the number k of tests and number t of messages that need to be corrupted for successfully cheating both scale linearly with the total number s of rounds and let $\kappa := \frac{k}{s}$ and $\alpha := \frac{t}{s-k}$ be the corresponding fractions. Let $\beta := \frac{\alpha}{1-\alpha}$ and $\gamma := (1-\alpha)^{\frac{1}{\alpha}}$. Then the optimal fraction of tests is given by $\kappa^*_\alpha := \frac{1-\gamma}{1+\beta\gamma}$ and the optimal cheating probability satisfies $\epsilon_S = \mathcal{O}((1+\beta\gamma)^{-s})$.*

Proof. The optimal attack of the Adversary is to corrupt exactly t messages out of s, since this is the minimal amount that is required for cheating successfully and corrupting more only increases the probability of getting caught. The probability that the attack succeed (which is also the probability that

[8] In stand-alone models this step of the Simulator uses rewinding to make sure that the Adversary has these proofs because it is it's only way of recovering them, but it is not necessary here.

the Simulator fails in the proof against malicious Sender by outputting Fail) is therefore upper-bounded as follows:

$$\Pr[\mathsf{Fail}] \leq \frac{\binom{s-t}{k}}{\binom{s}{k}} = \frac{(s-t)!}{k!\,(s-t-k)!} \cdot \frac{k!\,(s-k)!}{s!} = \frac{(s-t)!\,(s-k)!}{(s-t-k)!\,s!} \tag{4.21}$$

Using Stirling's formula we get that:

$$\frac{(s-t)!\,(s-k)!}{(s-t-k)!\,s!} \sim \sqrt{\frac{(s-t)(s-k)}{(s-t-k)s}}\,\frac{(s-t)^{s-t}(s-k)^{s-k}}{(s-t-k)^{s-t-k}s^s}e^{-s+t-s+k+s-t-k+s} \tag{4.22}$$

We now use the fact that k and t are linear in s and can therefore be written as $k = \kappa s$ and $t = \tau s$, simplifying by s to get:

$$\sqrt{\frac{(1-\tau)(1-\kappa)}{1-\tau-\kappa}}\left(\frac{(1-\tau)^{1-\tau}(1-\kappa)^{1-\kappa}}{(1-\tau-\kappa)^{1-\tau-\kappa}}\right)^s \tag{4.23}$$

Alternatively, since the corrupted messages are a fraction of the non-tested messages, we have $\tau = \alpha(1-\kappa)$, which allows us to simplify the quantity above though the following steps (without loss of generality we assume $\alpha \in (0,1)$):

$$\begin{aligned}
&= \sqrt{\frac{1-\alpha+\alpha\kappa}{1-\alpha}}\left(\frac{(1-\alpha+\alpha\kappa)^{1-\alpha+\alpha\kappa}(1-\kappa)^{1-\kappa}}{[(1-\alpha)(1-\kappa)]^{(1-\alpha)(1-\kappa)}}\right)^s \\
&= \sqrt{1+\frac{\alpha}{1-\alpha}\kappa}\left(\left(1+\frac{\alpha}{1-\alpha}\kappa\right)^{1-\alpha+\alpha\kappa}(1-\kappa)^{\alpha(1-\kappa)}(1-\alpha)^{\kappa}\right)^s \\
&= \sqrt{1+\frac{\alpha}{1-\alpha}\kappa}\left(\left(1+\frac{\alpha}{1-\alpha}\kappa\right)^{1-\alpha}(1-\kappa)^{\alpha}\left((1-\alpha)^{\frac{1}{\alpha}}\frac{1+\frac{\alpha}{1-\alpha}\kappa}{1-\kappa}\right)^{\alpha\kappa}\right)^s \\
&= \sqrt{1+\frac{\alpha}{1-\alpha}\kappa}\left(\left(1+\frac{\alpha}{1-\alpha}\kappa\right)^{\frac{1-\alpha}{\alpha}}(1-\kappa)\left((1-\alpha)^{\frac{1}{\alpha}}\frac{1+\frac{\alpha}{1-\alpha}\kappa}{1-\kappa}\right)^{\kappa}\right)^{\alpha s}
\end{aligned} \tag{4.24}$$

Considering α as a fixed parameter given in advance of the protocol,[9] we wish to minimise the following function of κ, i.e. find the optimal number of tests that the honest Receiver needs to perform:

$$f_\alpha(\kappa) := \left(1+\frac{\alpha}{1-\alpha}\kappa\right)^{\frac{1-\alpha}{\alpha}}(1-\kappa)\left((1-\alpha)^{\frac{1}{\alpha}}\frac{1+\frac{\alpha}{1-\alpha}\kappa}{1-\kappa}\right)^{\kappa} \tag{4.25}$$

We rewrite this using the parameters $\beta := \frac{\alpha}{1-\alpha}$ and $\gamma := (1-\alpha)^{\frac{1}{\alpha}}$:

$$f_\alpha(\kappa) := (1+\beta\kappa)^{\frac{1}{\beta}}(1-\kappa)\left(\gamma\frac{1+\beta\kappa}{1-\kappa}\right)^{\kappa} \tag{4.26}$$

[9] This is essentially the maximum amount of errors that the protocol which uses as a subroutine can correct across the received sets before being itself corrupted.

Let $A(\kappa) := (1+\beta\kappa)^{\frac{1}{\beta}}$, $B(\kappa) := (1-\kappa)$ and $C(\kappa) := \left(\gamma\frac{1+\beta\kappa}{1-\kappa}\right)^{\kappa}$, such that $f_\alpha(\kappa) = A(\kappa)\cdot B(\kappa)\cdot C(\kappa)$. Taking the derivative with respect to κ gives:

$$\begin{aligned}\frac{\mathrm{d}A(\kappa)}{\mathrm{d}\kappa} &= \frac{A(\kappa)}{1+\beta\kappa}\\ \frac{\mathrm{d}C(\kappa)}{\mathrm{d}\kappa} &= C(\kappa)\cdot\Big[\ln\Big(\gamma\frac{1+\beta\kappa}{1-\kappa}\Big)+\frac{(\beta+1)\kappa}{(1+\beta\kappa)(1-\kappa)}\Big]\end{aligned} \tag{4.27}$$

Then:

$$\begin{aligned}\frac{\mathrm{d}f_\alpha(\kappa)}{\mathrm{d}\kappa} &= \frac{ABC(\kappa)}{1+\beta\kappa} - AC(\kappa) + ABC(\kappa)\Big[\ln\Big(\gamma\frac{1+\beta\kappa}{1-\kappa}\Big)+\frac{(\beta+1)\kappa}{(1+\beta\kappa)(1-\kappa)}\Big]\\ &= AC(\kappa)\left(\frac{1-\kappa}{1+\beta\kappa}-1+\frac{(\beta+1)\kappa}{1+\beta\kappa}\right)+ABC(\kappa)\ln\Big(\gamma\frac{1+\beta\kappa}{1-\kappa}\Big)\\ &= ABC(\kappa)\ln\Big(\gamma\frac{1+\beta\kappa}{1-\kappa}\Big)\\ &= f_\alpha(\kappa)\ln\Big(\gamma\frac{1+\beta\kappa}{1-\kappa}\Big)\end{aligned} \tag{4.28}$$

Since $f_\alpha(\kappa) > 0$ for $\kappa\in(0,1)$, we have:

$$\frac{\mathrm{d}f_\alpha(\kappa)}{\mathrm{d}\kappa} > 0 \Leftrightarrow \gamma\frac{1+\beta\kappa}{1-\kappa} > 1 \Leftrightarrow \kappa > \frac{1-\gamma}{1+\beta\gamma} \tag{4.29}$$

The optimal value is therefore $\kappa^*_\alpha = \frac{1-\gamma}{1+\beta\gamma}$. Then, replacing κ^*_α yields (with $\gamma\frac{1+\beta\kappa^*_\alpha}{1-\kappa^*_\alpha}=1$):

$$\begin{aligned}f_\alpha(\kappa^*_\alpha)^\alpha &= (1+\beta\kappa^*_\alpha)^{1-\alpha}(1-\kappa^*_\alpha)^\alpha\\ &= \Big(1+\beta\frac{1-\gamma}{1+\beta\gamma}\Big)^{1-\alpha}\Big(1-\frac{1-\gamma}{1+\beta\gamma}\Big)^{\alpha}\\ &= \frac{(1-\alpha)(1+\beta)}{1+\beta\gamma}\\ &= \frac{1}{1+\beta\gamma}\end{aligned} \tag{4.30}$$

Recall that the cheating probability $\Pr[\mathsf{Fail}] = \mathcal{O}(f_\alpha(\kappa)^{\alpha s})$. For a fixed value of α and $\kappa^*_\alpha s$ checks, this gives directly $\Pr[\mathsf{Fail}] = \mathcal{O}((1+\beta\gamma)^{-s})$. This concludes the proof.

■

If the Adversary needs to corrupt at least $\alpha = 1/2$ of the forwarded sets,[10] we can plug this into our formulae above and get that $\beta = 1$, $\gamma = 1/4$ and therefore $\kappa^*_\alpha = 3/5$. This is highly counter-intuitive and all works up to now always use half of the sets as tests, i.e. $\kappa = 1/2$. This shows once again that the folkloric approach is not always optimal. All previous works that fit into this framework automatically obtain a better bound simply by switching to this new value of κ^*_α.

We can also analyse what happens for other values of α (for example by not using a majority vote but some other error-correcting technique). The value that governs the failure probability is $\beta\gamma$, which we wish to maximise. We can therefore analyse the influence of α on this value.

[10] For example in the case where the outer protocol uses a majority vote to error-correct possible deviation in the sets. This is used in [88] to secure Yao's Protocol against malicious Garblers.

Corollary 4.1 (Influence of Error-Correcting Procedure)**.** *Finding a better error-correcting procedure yields a lower maximum cheating probability.*

Proof. The bound on the cheating probability is given by $(1+\beta\gamma)^{-s}$. Minimising this for a fixed value of s requires us to maximise $\beta\gamma$. We therefore now consider $\beta(\alpha) = \frac{\alpha}{1-\alpha}$ and $\gamma(\alpha) = (1-\alpha)^{\frac{1}{\alpha}}$ as functions of α. Let $g(\alpha) := \beta(\alpha)\cdot\gamma(\alpha)$. Taking the derivative with regard to α, we have:

$$\begin{aligned} \frac{\mathrm{d}\beta(\alpha)}{\mathrm{d}\alpha} &= \frac{1}{(1-\alpha)^2} \\ \frac{\mathrm{d}\gamma(\alpha)}{\mathrm{d}\alpha} &= \gamma(\alpha)\Big(\frac{-\ln(1-\alpha)}{\alpha^2} - \frac{1}{\alpha(1-\alpha)}\Big) \end{aligned} \tag{4.31}$$

Then:

$$\begin{aligned} \frac{\mathrm{d}g(\alpha)}{\mathrm{d}\alpha} &= \frac{\gamma(\alpha)}{(1-\alpha)^2} + \beta(\alpha)\gamma(\alpha)\Big(\frac{-\ln(1-\alpha)}{\alpha^2} - \frac{1}{\alpha(1-\alpha)}\Big) \\ &= \gamma(\alpha)\Big(\frac{1}{(1-\alpha)^2} - \frac{\ln(1-\alpha)}{\alpha(1-\alpha)} - \frac{1}{(1-\alpha)^2}\Big) \\ &= -\gamma(\alpha)\frac{\ln(1-\alpha)}{\alpha(1-\alpha)} \end{aligned} \tag{4.32}$$

Since $\gamma(\alpha) > 0$, $\ln(1-\alpha) < 0$ and $\alpha(1-\alpha) > 0$ then $g'(\alpha) > 0$ so g is strictly increasing.

■

This means that forcing the Adversary to corrupt proportionally more circuits by looking for ways to perform error-correction that go beyond the majority vote on the output would automatically improve any scheme that relies on exponentially-secure classical CC.

4.3.3 Discussion

In is important to keep in mind when using FC-CC as a subroutine in another protocol that it only guarantees that a fraction of evaluation sets have been correctly generated. Any decision that is not based on this fact after the sets have been transmitted may easily lead to an attack that scales polynomially with the number of checks. While it might seem obvious after the security properties of FC-CC have been formalised, we list here the additional a few of the attacks found previously that are based on a misuse of this functionality. Details on these issues can be found in [88, 80] in the case of the Classical Yao Protocol. There, the Evaluator decrypts $s/2$ garbled circuits and return the majority outcome at the end.

If multiple circuits are evaluated, the first problem is that of *input consistency*. It would seem that if only one final result is ever sent back there shouldn't be a problem if there are different inputs. To illustrate the importance of this concern, consider the function outputting the scalar product of the inputs in $\mathbb{F}_2^n$. If there are n evaluation circuits and the adversarial Garbler sends as inputs the n strings $(10\ldots0), (010\ldots0), \ldots, (0\ldots01)$ then the majority output will reveal the majority bit of the Evaluator's input. On the other hand, if the Evaluator is capable of doing so it can learn all of the Garbler's input.

Then, another issue has been pointed out in [80]. Consider the following process: the Adversary constructs $s-1$ correct circuits and one malicious circuit which returns the correct result if the first input bit of the Evaluator is 0 and outputs the binary complement of the correct output if it is 1. If the Evaluator's first input is 1 then this circuit will never be the majority circuit if it is evaluated. When the Evaluator's input is 0 then there is a $2/s$ probability that it is returned as the majority circuit if there are $s/2$ evaluation circuits. The probability of not having checked this circuit is $1/2$. So there is in total a $1/s$ probability that this circuit will appear as the majority circuit if the Evaluator's first input is 0. If this circuit appears, the Garbler knows that this is the case, therefore breaking the protocol's security. The takeaway message is that no message should reveal the index of the majority circuit.

Finally, another problem that appears is caused by the fact that the Evaluator, if it is given a wrong input key through the OT, will be forced to abort because it will be unable to decrypt the circuit (in Yao's Protocol an incorrect decryption can be detected by the Evaluator). The *Selective OT Attack* works this way: for the Evaluator's first input, the Garbler sends the correct key for the value 0 and a wrong key for the value 1. Then, depending on whether the Evaluator aborts or not, the Garbler learns its first input bit. Furthermore, the Evaluator has no idea that any information leaked. This attack is thwarted classically by modifying the circuit and introducing $\mathcal{O}(m)$ new gates (m is the length of the inputs).

Coin-Tossing for Simulation. The coin tossing resource might appear to be superfluous as it would seem as though the Receiver should be able to choose on their own the check sets without any impact on the security of the protocol. In fact we do not expect the same protocol without the coin-tossing protocol to be subject to any attack, but the Simulator for malicious Receiver has to be able to force the Adversary to choose a specific subset. While this could be done via the oblivious rewinding technique in the previous section, it is not applicable here since the probability of succeeding in the simulation must not be negligible for this technique to work. Here the probability that the simulator and Adversary choose the same subset is precisely negligible. This is the only reason why a coin-tossing protocol is required.[11]

On a related note, the reason why the security proofs above are much simpler than in the previous section is that we do not use rewinding since it is incompatible with composable frameworks such as AC. However, we do require two new primitives and a much stronger extractability requirement than previously. In particular, this imposes that the proof for any given correct message is not only unique but also that there is a procedure to efficiently extract it using the trapdoor.

Constant Factor of Failure Probability. In order to get a more precise bound and find the factor hidden by the order notation, we can bound the factorials in the proof of Lemma 4.4 using the following inequality from [119] (or using more terms in Stirling's approximation):

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} < n! < \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}} \tag{4.33}$$

[11] Note that the proof against malicious Sender will only require that this subset be random and the Simulator does not need to influence it whatsoever. In the stand-alone setting, this means that the coin-tossing protocol could be only simulatable for one side and random for the other. Such protocols are much easier to create, using only commitment schemes, compared to those which are simulatable for both parties. However, in a fully composable scenario such as the AC framework, all protocols must be simulatable for all parties.

It can easily be checked from equation 4.24 that this factor decreases and approaches the following constant as s increases:

$$\sqrt{1+\beta\kappa_\alpha^*} = \sqrt{\frac{1+\beta}{1+\beta\gamma}} \tag{4.34}$$

4.4 The Protocol Compiler

We now present a Compiler which boosts the security of a protocol between parties P_1 (acting as the Sender) and P_2 (as the Receiver) from a very weak adversarial model called Semi-Malicious to the inverse-polynomially Malicious setting for P_1, while preserving all of its other security guarantees. The goal of the Compiler is not only to guarantee the fact that P_1 has sent a correctly prepared state and message but also that future interactions depending on these are also correct.

High-Level Overview of the Compiler. We suppose that the protocol to be compiled (called an Abstract Protocol) can be decomposed in three steps: a first interaction, followed by one party sending a CC-able state $|\psi\rangle$ and message m and finally a second interaction. The goal is to remove the opportunity for an adversarial P_1 to dishonestly prepare and send messages in this later interaction (let m' be such a message) by leveraging the same Q-CC procedure as the one that is used to secure the CC-able state and message. However, at the time when the Q-CC is performed, these messages may either not be know to P_1 because they depend on a message that is sent by P_2 in a later round, or they may depend on a secret parameter of P_1. In both cases however, if the set from which these messages or parameters are chosen is known to P_1, it can pre-compute the value of the message m' to be secured by iterating over all possible values of the messages that m' depends on (if this is efficient) and send to P_2 a commitment to these pre-computed values (this commitment does not reveal information until opened). During the Q-CC procedure, P_2 can check that these have been pre-computed correctly using the opening information of the commitments (which are appended to the proof). For the unchecked set, a subset of these messages can be revealed selectively through leaks in the second interaction. The construction of these sets must however follow certain rules for them to preserve the security of both players. Given these pre-computed sets, the Compiler works by making the parties run the first protocol multiple times, then performing a Q-CC procedure which includes these pre-computed sets and continuing the execution on the unchecked instance, replacing any message sent by the Sender from the secured set by the opening of the corresponding commitment.

We first introduce a new weak Adversary called Semi-Malicious (as mentioned in the Introduction, these Adversaries are not related to other definitions of Semi-Malicious parties). Proving the security of a protocol against this adversarial model is simpler than for Malicious Adversaries while our Compiler will make sure that such a protocol may be strengthened into one which resists arbitrary Malicious parties.

4.4.1 New Semi-Malicious Adversaries

The Semi-Malicious Adversary may deviate in any way it wants except when preparing and sending a specific subset of the messages of the protocol, for which it must act as an honest player would.

Definition 4.11 (Semi-Malicious Adversary)**.** *Let* Π *be a two-party protocol and* $\mathcal{M}_{SM}$ *a publicly known subset of the* classical *messages sent by party* P_i*, in order of appearance. We say that an Adversary* $\tilde{\mathcal{A}}$ *controlling party* P_i *is* $\mathcal{M}_{SM}$*-Semi-Malicious (or* $\mathcal{M}_{SM}$*-SM) if* P_i *prepares and sends honestly all messages in* $\mathcal{M}_{SM}$ *during the execution of the protocol. It can be arbitrarily Malicious in all the other steps of the protocol.*

The honest preparation for a subset of messages in a protocol is defined in the following sense. It means that, given the messages previously sent in the protocol, the message is prepared using the correct function applied to these anterior messages using, if required, randomness sampled in the correct set (we do not enforce the correctness of the distribution used in this sampling). Furthermore, if two messages depend on the same randomness or secret parameters, then the same values for these are used in the generation of both messages.

Definition 4.12 (Computational Security against Semi-Malicious Adversary)**.** *We say that a two party protocol* Π $\epsilon(n)$*-securely emulates Ideal Functionality* $\mathcal{F}$ *against* $\mathcal{M}_{SM}$*-SM Adversaries if it is* $\epsilon(n)$*-secure according to Definition 3.7 when quantifying over all quantum polynomial-time* $\mathcal{M}_{SM}$*-SM Adversaries.*

The adversarial model defined here may be very weak depending on the subset $\mathcal{M}_{SM}$ that is chosen and is not to be considered in any real-world protocol. On the other hand the security proofs against these Adversaries are much simpler, as shown in Section 4.5. The Compiler presented below transforms any protocol secure against these weak Adversaries into one secure against Malicious Adversaries by enforcing the honest behaviour of the Adversary on messages in the set $\mathcal{M}_{SM}$.

4.4.2 Constraints on Abstract Protocols

The Compiler (Transformations 1 and 2) is applicable to protocols following the structure of the Abstract Protocol 9 (or AP), emulating some Ideal Functionality $\mathcal{F}$, where $S_{\mathscr{C}}$ is a CC-able CP-map (together with the associated $R_{\mathscr{C}}$).

A more general protocol would allow for some messages in Π^1 to depend on the message and proof of the CC-able set (i.e. the set is generated during Π^1 but sent at the end). In some cases it may be possible to rewrite Π^1 and the CC-able CP-maps $S_{\mathscr{C}}$ and $R_{\mathscr{C}}$ to sample these messages otherwise and adapt the state, message and proof so that they are coherent (by including these dependent messages in input *inp* for instance). It would then be possible to recover the AP structure presented above, however we do not consider such an anachronistic state generation technique here (we do allow it for Simulators on the other hand, as described below).

For the transformation to be applicable to an AP, there needs to be further restrictions on its structure and we call CC-able Protocol (Definition 4.13) an AP that satisfies the three criteria given below. The first one is required for the correctness of the Compiler to hold, while the other two imply its security properties.

Definition 4.13 (Cut-and-Choosable Protocol)**.** *We say that a secure AP (i.e. following the structure of Protocol 9) between parties* P_1 *and* P_2 *with the set* $\mathcal{M}_{SM}$ *and round* r^{CC} *defined as above is a* Cut-and-Choosable Protocol *if:*

- *It is* pre-computable *for set* $\mathcal{M}_{SM}$ *at round* r^{CC} *(Definition 4.15) for honest* P_1 *and* $Sim_{P_2^*}$ *(the Simulator against Malicious* P_2^**);*

Protocol 9 Abstract Protocol

Public Information: Public string $\widetilde{\mathscr{C}}$, Semi-Malicious secure set $\mathcal{M}_{SM}$.
Inputs: P_1 and P_2 have inputs $x \in \{0,1\}^n \cup \{\lambda\}$ and $y \in \{0,1\}^n \cup \{\lambda\}$ respectively.
Protocol:

1. P_1 and P_2 perform an interaction $\Pi^1[x,y]$, yielding internal states $\left(\rho_1^1, \rho_2^1\right)$ and classical information (aux_1^1, aux_2^1) for P_1 and P_2. The public string $\widetilde{\mathscr{C}}$ may also be modified by appending additional public knowledge during this interaction, resulting in public string $\mathscr{C}$. Furthermore, at the end of Π^1, P_1 is in possession of $inp \in \mathbf{I}$, whether it is included in its input x or generated during the execution and therefore contained in aux_1^1.
2. P_1 samples $sk \longleftarrow D_{sk}(\eta, \mathscr{C})$ and applies $S_{\mathscr{C}}$, obtaining $(\mathcal{X}, m_{sk}, proof_{sk}, info_{sk})$. It sends the set $(\mathcal{X}, m_{sk})$ to P_2. Let r^{CC} be the round at which this step is performed.
3. P_1 and P_2 perform an interaction $\Pi^2\left[\left(x, \rho_1^1, aux_1, m_{sk}, info_{sk}\right), \left(y, \rho_2^1, aux_2, |\psi_{sk}\rangle, m_{sk}\right)\right]$, yielding internal states (ρ_1^2, ρ_2^2), classical information (aux_1^2, aux_2^2) and outputs (out_1, out_2) for P_1 and P_2. We suppose that all messages $M \in \mathcal{M}_{SM}$ are sent by P_1 after round r^{CC}. During this interaction, P_1 may be required to sends parts of $proof_{sk}$ to P_2. The rounds at which this happens and the parts of the proof that are sent are specified by Π^2 in the following way: to each $leak \in \mathcal{L}_{\mathscr{C}}$ we associate r^{leak}, corresponding to the round at which $leak$ may be requested (this is part of the leak-test algorithm $L_{\mathscr{C}}$). Note that two or more elements of $\mathcal{L}_{\mathscr{C}}$ may share the same round, meaning that both leaks may be queried simultaneously. We treat them nevertheless as two distinct calls.

- *The Simulators for both players are* CC-Compatible *(Definition 4.16);*
- *The Simulator for Malicious P_2^* is* CC-Specious *(Definition 4.17).*

Recall that the principle of the Compiler to force P_1 to send honestly messages from set $\mathcal{M}_{SM}$ by pre-computing all possible values for those messages that it is susceptible to send to P_2 during the execution of the protocol. For the sake of correctness, P_1 must be able to do so efficiently (in deterministic polynomial-time).

We start by formally defining dependent messages, i.e. one message is dependent on another if the value of the second message is required for computing that of the first one. Unless specified, in the following we consider formal messages and not the values of messages sent in an actual execution.

Definition 4.14 (Classically Dependent Messages)**.** *Fix a round r_d in the execution of a given two-party protocol Π. Let m be a classical message sent during the execution of protocol Π. We say that classical message m' sent by party P_1 at round $r' > r_d$ in protocol Π depends on message m (and write $m \to m'$)*[12] *if the value of the message m is needed to compute the value of message m' in a given honest execution of the protocol, i.e. there exist formal classical messages $(m_1, \ldots, m_j)$ and a classical deterministic function g such that it is specified in the protocol that $m' = g(m, m_1, \ldots, m_j)$ is sent by P_1 in round r'. We further distinguish the following types of dependencies (these implicitly depend on the round r_d that is being considered):*

- *Type 0: The value of formal message m is known to both parties at round r_d of the protocol (and thus also r').*[13]

[12] In any correct protocol, the dependencies of messages may be visualised as a Directed Acyclic Graph.
[13] This can be either a constant defined by the protocol or any message sent before round r_d.

- *Type 1: The value of m is not known yet to P_1 at round r_d of the protocol but will be sent by another party*[14] *in a future intermediate round r with $r' > r > r_d$.*
- *Type 2: The value of m is known to P_1 but not P_2 at round r' of the protocol.*[15]
- *Type 3: The value of m is sent by P_1 to P_2 at an intermediate round r with $r' > r > r_d$.*

We then also define T_i, the sets of messages of type i. This is well-defined since the type of each message is dependent only on the protocol, the party considered and the round r_d (i.e. it does not vary across message dependencies).[16]

Let r_d be a round of AP Π and τ_Π be the transcript up to round r_d of the execution of Π for which the messages are being pre-computed. Let $\hat{m}$ be a message in Π, we define the ordered set $\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi)$ recursively in the following way for round r_d (where π is a permutation on the set):

- All sets that are computed are kept in memory until every message from $\mathcal{M}_{SM}$ has been pre-computed. If the set for message $\hat{m}$ has already been computed then it is reused, along with its associated permutation. This guarantees consistency across messages that are revealed (if two messages both depend on the same message, then the Receiver is able to check that the position of revealed messages is coherent, meaning that the underlying value of this common dependent message is the same).
- If $\hat{m}$ is of type 0, let $\hat{m}(\tau_\Pi)$ be its value in this specific execution of the protocol. Then $\pi = Id_1$ (identity permutation on a single element) and $\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi) = \{\hat{m}(\tau_\Pi)\}$ as there is only one possible value and it is already known to both players.
- If $\hat{m}$ is of type 1 (it will be sent at a round after r_d by a party other than P_1), let $\hat{\mathbf{m}}(\tau_\Pi)$ be the ordered set of all possible values of $\hat{m}$ given the values τ_Π, then $\pi = Id_{\hat{\mathbf{m}}}$ and $\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi) = \hat{\mathbf{m}}(\tau_\Pi)$. This allows the Receiver to check that, once it does send $\hat{m}$, that the messages revealed that depend on $\hat{m}$ have been computed using the correct value of $\hat{m}$ (based on its position in the set).
- If $\hat{m}$ is of type 2 or 3 with its value being chosen by P_1 from a finite set $\hat{\mathbf{m}}(\tau_\Pi)$ (regardless of the distribution used):
  - If revealing the value of $\hat{m}$ in an execution of the protocol at round r_d breaks the security of a concurrent execution (by giving a non-negligible distinguishing advantage to the Environment if used as side-information), then P_1 chooses a random permutation π over $\hat{\mathbf{m}}(\tau_\Pi)$ and computes $\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi) = \pi(\hat{\mathbf{m}}(\tau_\Pi))$.
  - On the other hand, if revealing the value of message $\hat{m}$ does not decrease the security of any concurrent execution by more than a negligible amount $\epsilon_{\hat{m}}$, P_1 chooses a value $\hat{m}(\tau_\Pi) \in \hat{\mathbf{m}}(\tau_\Pi)$ (according to whichever distribution it prefers if Malicious). Then $\pi = Id_1$ and $\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi) = \{\hat{m}(\tau_\Pi)\}$. Note that by definition, the message m_{sk} generated by the state generation CP-map is of this type.
- If $\hat{m}$ is of type 3 and computed by applying classical deterministic polynomial-time function g on messages $(m_1, \ldots, m_k)$, then first the sets $\mathcal{S}_{m_l}(\tau_\Pi, \pi_m)$ for all $l \in [k]$ are computed. [17] Then,

[14] Either P_2 or a trusted third party in a hybrid execution.
[15] This case represents secret parameters of P_1 for example.
[16] Note that the type is known to both players.
[17] Or equivalently, messages that are anterior to others according to a topological ordering of the Directed Acyclic Graph defined by message dependencies are pre-computed first.

iterating in a predefined order (lexicographic for example, but known to both players) over all previously computed sets $\mathcal{S}_{m_l}(\tau_\Pi, \pi_l)$, the following ordered set is constructed (the permutation $\pi_{\hat{m}}$ being defined implicitly by the previously chosen π_l):

$$\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi_{\hat{m}}) = \{g(\tilde{m_1}, \ldots, \tilde{m_k}) \mid \forall l \in [k], \tilde{m_l} \in \mathcal{S}_{m_l}(\tau_\Pi, \pi_l)\} \tag{4.35}$$

Let $\mathcal{M}_{\mathfrak{GM}}$ be the set of messages $\hat{m}$ for which the set $\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi_{\hat{m}})$ has to be constructed during the pre-computation of messages in $\mathcal{M}_{SM}$. The decrease in security $\epsilon_{\hat{m}}$ is modelled as the maximum difference in distinguishing advantage for any Distinguisher when its auxiliary input contains the value of the message $\hat{m}$ for a previous execution of the protocol and when it doesn't, taken over all possible executions. Let $\mathcal{M}_{SM}^{fix} \subseteq \mathcal{M}_{\mathfrak{GM}}$ be the set of messages fixed during pre-computation of messages from set $\mathcal{M}_{SM}$. We call $\epsilon_{pre-c} := \sum_{\hat{m} \in \mathcal{M}_{SM}^{fix}} \epsilon_{\hat{m}}$ the upper-bound on the total decrease in security for the leakage of the messages in $\mathcal{M}_{SM}^{fix}$ for any possible execution (i.e. the negligible advantage of the Adversary in attacking an execution of a protocol given a pre-computed set of another execution is ϵ_{pre-c}).

Note that the same value for message M may appear multiple times in these sets if it appears in multiple branches of the protocol execution, e.g. : if $M = f(m, r)$ for some function f and messages (m, r), and for the chosen value $\tilde{m}$ of formal message m there exists value r_1 and r_2 permissible by the protocol such that $M_1 = f(\tilde{m}, r_1) = M_2 = f(\tilde{m}, r_2)$, that value of message M will appear twice in the set $\mathcal{S}_M(\tau_\Pi, \pi_M)$.

We can then define the following set (where π is a permutation defined implicitly by the ones chosen for the pre-computed set of each message in $\mathcal{M}_{\mathfrak{GM}}$):

$$\begin{aligned} \Xi^{P_i}_{\mathcal{M}_{SM}}(\tau_\Pi, \pi) := & \left\{\pi_{\hat{m}} \mid \hat{m} \in \mathcal{M}_{\mathfrak{GM}} \cap T_2 \setminus \mathcal{M}_{SM}^{fix}\right\} \cup \\ & \left\{\hat{m}(\tau_\Pi) \mid \hat{m} \in \mathcal{M}_{SM}^{fix}\right\} \cup \\ & \{\mathcal{S}_{\hat{m}}(\tau_\Pi, \pi_{\hat{m}}) \mid \hat{m} \in \mathcal{M}_{\mathfrak{GM}} \cap T_3\} \end{aligned} \tag{4.36}$$

It corresponds to the set of all parameters associated to $\mathcal{M}_{SM}$ that result from a choice of player P_1. An equivalent set would be obtained by first imposing that the set $\mathcal{M}_{SM}$ is closed under the pre-computation procedure, i.e. $\mathcal{M}_{\mathfrak{GM}} = \mathcal{M}_{SM}$.

A protocol must then satisfy the following property (for P_1) in order to be correctly compilable. To preserve the security against adversarial P_2, the corresponding Simulator (that acts as P_1) must also satisfy it.[18] Note that this requirement imposes that not only the messages in $\mathcal{M}_{SM}$ must be classical, but the messages they depend on must be as well.

Definition 4.15 (Pre-computable Protocol)**.** *Let (ρ_1^1, aux_1) be the internal state of the honest Sender after execution of sub-protocol Π^1. We say that AP Π is pre-computable for P_1 for set $\mathcal{M}_{SM}$ at round r^{CC} if there exists an efficient CP-map $PreC_{aux_1}$ such that $PreC_{aux_1}(\rho_1^1) = \left(\tilde{\rho}_1^1, \Xi^{P_1}_{\mathcal{M}_{SM}}(\tau_{\Pi^1}, \pi)\right)$ and the composed protocol $\Pi^1 \circ PreC_{aux_1} \circ \Pi^2$ is as correct as Π.*[19]

[18] This is a stronger requirement than necessary since this Simulator may have a better strategy by faking part of the pre-computation and forcing the Adversary to pick the correct branch.

[19] i.e. The pre-computation does not decrease correctness. This can be easily be relaxed to tolerate a negligible decrease in correctness stemming from the pre-computation.

Now that correctness has been dealt with, the next two conditions are required for the Compiler to preserve the security of P_1 and boosts that of P_2. In order to prove that these properties are satisfied, the Simulators for the compiled protocol must be able to reuse the ones for the non-compiled version.

A first necessary condition for this to be possible is that the input of the Adversary in the compiled protocol must be well-defined when the Simulator performs the call to the Ideal Functionality. Otherwise the Simulator for the compiled protocol would not know on which of the s executions of protocol Π^1 to call the single-query oracle. This is formalised below in Definition 4.16.

Definition 4.16 (CC-Compatible Simulations)**.** *Let Π be an AP securely emulating an Ideal Functionality $\mathcal{F}$. The simulators constructed while proving its security are said to be* CC-compatible *if either of the two following conditions are satisfied (each Simulator may satisfy a different condition):*

- *The Simulator performs the call to the single-query oracle implementing Ideal Functionality $\mathcal{F}$ during the execution of sub-protocol Π^2.*
- *There exists an extension to protocol Π^1 called Ext$-\Pi^1$ in which the probability $\epsilon_{cc-fail}$ that the input of the Adversary in one of the s executions of protocol Ext$-\Pi^1$ is different from that used in another is negligible.*

There are multiple options for satisfying the second condition. One can extend the protocol using Zero-Knowledge Proofs that there exists an input consistent across all executions. Another way would be to specify that the Adversary must use identical messages for all s executions. Both of these have drawbacks: the Zero-Knowledge Proofs add an overhead on the complexity of the protocol[20], while forcing a player to use the same message for all executions could lead to a decrease in security. See the proofs of security of the protocol presented in Section 4.5 for more concrete examples for how to satisfy these conditions.

One last condition must be imposed on the Simulator against Malicious P_2. It not only needs to be able to pre-compute its messages for the evaluation set, but must also be able to create check sets that, once opened, pass verification and concord with the Simulator's sent messages. These checks sets can be "faked" by the Simulator using the CP-map defined in Definition 4.17 below.[21]

Definition 4.17 (CC-Specious Simulation)**.** *Let Π be an AP secure against Malicious P_2^* and $Sim_{P_2^*}$ the associated Simulator. Let $\tau(\Pi^1, Sim_{P_2^*})$ be the transcript between the Adversary and the Simulator during the execution of protocol Π^1 and $\rho_{Sim_{P_2^*}}$ the internal state of the Simulator at the end of the execution of protocol Π^1. The simulation is said to be* CC-specious *if there exists a negligible $\epsilon_{SS}(\eta)$ along with a CP-map $S^*_{\mathscr{C}}\left(\tau(\Pi^1, Sim_{P_2^*}), \rho_{Sim_{P_2^*}}\right)$ producing $(\mathcal{X}, m^*, proof^*)$ with $\mathcal{X}$ being a quantum register containing state $|\psi^*\rangle$, such that, for all Environments $\mathcal{Z}$ with auxiliary input register $\mathcal{W}_{\mathcal{Z}}$ and Adversary $\mathcal{A}$ controlling corrupted party P_2^*:*

$$\left|\Pr\left[1 \leftarrow \mathcal{Z}\big(v(Sim_{P_2^*}, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in}), \mathcal{W}_{\mathcal{Z}}\big)\right] - \Pr\left[1 \leftarrow \mathcal{Z}(v(P_1, \mathcal{A}(\rho_{\mathcal{A}}), \rho_{in}), \mathcal{W}_{\mathcal{Z}})\right]\right| \leq \epsilon_{SS}(\eta) \tag{4.37}$$

[20] It would also prove hard to use non-generic ZKPs for this purpose, which would then be equivalent to the GMW Compiler. This in turn, although still polynomial, is known for being highly inefficient. If going that route, one might as well use this Compiler for securing the set $\mathcal{M}_{SM}$ as well.

[21] Here however, there are not shortcuts such as the one described in Footnote 18 above since these will be checked and must pass the test.

In the formula above, the terms v are similarly defined as in Definition 3.7, with the difference that it takes into account only protocol Π^1 along with the revelation of the state, message and proof (in the honest case using the CP-map $S_{\mathscr{C}}$ while the Simulator uses $S^*_{\mathscr{C}}$). The Environment and Adversary are restricted to being QPT *machines in the computational case.*

Comments on CC-Specious Simulators Note that this is not linked directly to the definition for Specious Adversaries (as may be found in [44] for instance). It is however similar since it captures the fact that the Simulator is able to produce, if asked, a state, message and proof which verifies correctly and is coherent (and indistinguishable from honest) with the transcript sent previously.

This constraint is necessary if there exists a message m sent during protocol Π^1 that is related to the state, message and proof generated by the Sender CP-map $S_{\mathscr{C}}$. This message will necessarily be included in the input *inp* for the Sender's CP-map.[22] In the non-compiled protocol, the Simulator never needs to reveal how it generates the state and so it may do so without there being a valid proof associated to it.

We remark that if the key-indistinguishability property (Definition 4.5) is defined using a trap-door as mentioned in Footnote 5, then the CC-specious Simulation property (Definition 4.17) can be merged with it.

Both properties are also not equivalent to the protocol being sequentially or concurrently self-composable as it is required here that even revealing extra information, to which the Adversary does not have access in a regular execution, does not break a concurrent execution. It is however a weaker requirement than Quantum Universal-Composability [126] and therefore the Compiler is also applicable to protocols which are Q-UC. Unfortunately, since we use quantum rewinding in the proof, the compiled protocol is not Q-UC.

The Q-CC Protocol may be adapted to fit the Q-UC requirements if a trusted setup with trapdoor is added akin to what is done for commitment schemes, by strengthening the trapdoor to not only allow the simulator to generate fake check sets in Definitions 4.5 but also extract the proof from the message (by updating slightly Definition 4.2). By additionally using a Q-UC commitment scheme when constructing the pre-computed sets above, the full construction can be made Q-UC secure.

4.4.3 Presentation of the Compiler

We can now define our Compiler which transforms a protocol secure against Semi-Malicious Adversaries into one secure against Malicious Adversaries.

It will require P_1 to commit to messages using Bit Commitment $(\mathsf{Com}, \mathsf{Verif})$ before sending them to P_2. This primitive must be be perfectly correct (all commitments honestly generated with Com are verified by Verif as correct given the opening information), perfectly hiding (no unbounded Adversary with access to the commitment can infer information about the message used to generate it) and collapsing (given a valid commitment and associated message quantum register, no computationally-bounded Adversary can tell whether this register has been measured or not). The formal definitions for these properties can be found in Section 3.1.2.

The Compiler consists of two consecutive steps, the first one modifying the CC-able CP-maps and related information (Transformation 1) while the second one changes the outer protocol accordingly

[22] Otherwise this property is trivially verified with $\epsilon_{SS} = 0$ since the Simulator can then use the same CP-map as the honest player.

(Transformation 2). It takes as input a CC-able Protocol (Definition 4.13) and outputs a protocol that is inverse-polynomially secure against quantum polynomial-time Malicious P_1 while keeping all its other properties intact.

Multiple instances of the protocol Π will be run until the Q-CC round r^{CC} (in parallel if protocol Π_1 is self-composable in parallel, or sequentially otherwise). A commitment scheme is used to commit to sets of pre-computed messages in the order specified above, with the opening information being stored in the proof of a CC-able message. After the two players are done with performing the Q-CC Protocol, only the remaining unchecked version of the protocol will continue its execution, the others having only served for tests. If a given message is in the secured set $\mathcal{M}_{SM}$, instead of sending the message, the Sender must send the opening information for the corresponding commitment (as part of a leak).

Transformation 1 The Compiler: Step 1 - Modifying the CP-maps

Modifying $\mathscr{C}$: Each party appends $\tau^i_{\Pi^1}$ to $\mathscr{C}$, corresponding to all message exchanged (in order) during the execution of protocol Π^1 for each execution $i \in [s]$. Let $\mathscr{C}'$ be the resulting string.
Modifying $S_{\mathscr{C}}$:

1. $S'_{\mathscr{C}'}$ now takes as input an additional value i representing the execution index of the protocol.
2. $S'_{\mathscr{C}'}$ applies $S_{\mathscr{C}}(sk, inp^i)$ and obtains $(\mathcal{X}, m_{sk}, proof_{sk}, info_{sk})$.
3. Using $\tau^i_{\Pi^1}$, $S'_{\mathscr{C}'}$ computes the set $\Xi^{P_1}_{\mathcal{M}_{SM}}(\tau^i_{\Pi^1}, \pi^i)$ for execution i and appends it to m_{sk} (this includes the permutation and the fixed messages as well). Let $\tilde{m}_{sk}$ be the resulting message.
4. Using Com, $S'_{\mathscr{C}'}$ commits to each separate message of $\tilde{m}_{sk}$ apart from the original m_{sk}, we denote $m_{sk,COM}$ the resulting committed message.
5. It appends to $proof_{sk}$ the opening information to all the commitments constructed in the previous step, let $proof_{sk,OP}$ be the resulting message.
6. At the end $m_{sk,COM}$ contains only commitments and m_{sk}, while $proof_{sk,OP}$ only contains opening information and $proof_{sk}$.

Modifying $R_{\mathscr{C}}$ (if any step fails, it outputs 0):

1. $R'_{\mathscr{C}'}$ now takes as input an additional value i representing the execution index of the protocol.
2. Using the opening information in $proof_{OP}$ and Verif, $R'_{\mathscr{C}'}$ checks all commitments in m_{COM}.
3. Using $\tau^i_{\Pi^1}$, it checks that the set $\Xi^{P_1}_{\mathcal{M}_{SM}}(\tau^i_{\Pi^1}, \pi^i)$ is correctly constructed with regard to permutation π^i (note that, given the fixed choices for messages in $\mathcal{M}^{fix}_{SM}$, P_2 is able to produce the rest of this set on its own as it only correspond to all the possible values for the messages sent by the other player given the transcript of previous messages).
4. It parses the first part of m_{COM} and $proof_{OP}$ as m and $proof$ respectively and returns the output of $R_{\mathscr{C}}(\mathcal{X}, m, proof)$.

Modifying $\mathcal{L}_{\mathscr{C}}$: Let $leak_M$ be the set of indices in $proof_{sk,OP}$ corresponding to the opening information of all pre-computed values for message $M \in \mathcal{M}_{\mathfrak{SM}} \cap T_3$. Then $\mathcal{L}'_{\mathscr{C}} = \mathcal{L}_{\mathscr{C}} \cup \{leak_M \mid M \in \mathcal{M}_{\mathfrak{SM}} \cap T_3\}$. The leak-selection function $L_{\mathscr{C}}$ is updated to select one value of message M per such additional set at round r^M if the honest Sender should have sent that value of message M at this round and these round values are added to the sequence of values r^{leak} (which define when a leak can be requested). $L_{\mathscr{C}}$ returns $\perp$ otherwise on these elements.

Note that neither the Cut-and-Choose procedure nor the Compiler requires more quantum power than the initial protocol (and so can be implemented at no extra cost) apart from storing all the states between reception and verification. With an additional assumption of 1-out-of-s Oblivious Transfer[23], this can be reduced to storing only at most 2 states instead of s by making the Receiver choose before

[23]This functionality works by having the Receiver input an index α and the Sender input for each index the proofs for all other sets. This would be executed before the states are sent.

Transformation 2 The Compiler: Step 2 - Modifying the Protocol

The CC-able Protocol is modified as such:

1. P_1 and P_2 run s times the protocol Π^1 (or $Ext{-}\Pi^1$ if using the second condition for CC-compatible simulators in Definition 4.16). This can be done concurrently if the protocol is self-composable in parallel.
2. They participate in a Q-CC Protocol with CC-able CP-maps $S'_{\mathscr{C}'}$ and $R'_{\mathscr{C}'}$. P_1 samples $sk \longleftarrow D_{sk}(\eta, \mathscr{C})$. It then applies s times the CP-map $S'_{\mathscr{C}'}(sk, inp^i)$ and obtains the sets $\left\{\mathcal{X}^i, m^i_{sk,COM}, proof^i_{sk,OP}, info^i_{sk}\right\}_{i\in[s]}$. P_1 sends the states and messages, P_2 chooses an evaluation index α, P_1 sends the proofs, P_2 checks the proofs using $R'_{\mathscr{C}'}$, accepting or rejecting the remaining state and message depending on the outcome of the tests.
3. They then continue with the execution of protocol Π^2 on the α^{th} instance. During this execution, both parties continue to append any messages exchanged to $\mathscr{C}'$. When the protocol specifies that a given message $M \in \mathcal{M}_{\mathfrak{S}\mathfrak{M}} \cap T_3$ should be sent, P_2 request a leak from P_1 for the correct value of M (by sending the set of indices corresponding to the pre-computed set for message M), who replies with the corresponding opening information if the leak is accepted. P_2 then checks the opening of the commitment in $m^{\alpha}_{sk,COM}$ by using Verif. P_2 also checks that the position of the opened message is consistent with previous messages (if two messages are dependent on the same message, their position with regard to this message should be the same since the permutation used in computing it has been reused). The other leaks are treated as in the original protocol.

receiving the states which executions will be tested and acquiring the corresponding proofs in advance, allowing it to test them as soon as they are sent.

The next subsection presents the properties of the Compiler and their proofs, namely that it preserves the security properties of the CC-able Protocol while boosting its security from Semi-Malicious to fully Malicious.

4.4.4 The Compiler: Main Results

Since the Q-CC procedure is applied to CP-maps $S'_{\mathscr{C}'}$ and $R'_{\mathscr{C}'}$, we must also prove that these maps are indeed CC-able.

Lemma 4.5 (Transformed CC-able CP-maps)**.** *Let $S_{\mathscr{C}}$ and $R_{\mathscr{C}}$ be CC-able CP-maps with key-indistinguishability advantage ϵ_k and proof-collapsing advantage ϵ_c. Let $(\mathsf{Com}, \mathsf{Verif})$ be a perfectly-hiding and $\epsilon_c^{\mathsf{Com}}$-collapsing commitment scheme. Let ϵ_{pre-c} be the decrease in security against Malicious P_2^* stemming from revealing one pre-computed set.* [24] *Then $S'_{\mathscr{C}'}$ and $R'_{\mathscr{C}'}$ obtained after applying Transformation 1 CC-able CP-maps with key-indistinguishability advantage $\epsilon'_k = \epsilon_k + \epsilon_{pre-c}$ and proof-collapsing advantage $\epsilon'_c = \epsilon_c + \epsilon_c^{\mathsf{Com}}(M^{fix} + X_\Pi)$ where M^{fix} is the size of set $\mathcal{M}^{fix}_{SM}$ and X_Π is the size of the sets $\Xi^{P_1}_{\mathcal{M}_{SM}}(\tau^i_{\Pi^1}, \pi^i)$.*

Proof. Given that $S_{\mathscr{C}}$ and $R_{\mathscr{C}}$ are CP-maps, then $S'_{\mathscr{C}'}$ and $R'_{\mathscr{C}'}$ are trivially also CP-maps. It is straight-forward to show that they are extractable by re-using the extractor employed by $R_{\mathscr{C}}$ after the commitments have been opened. The new extractor outputs $\perp$ if a commitment has not been opened correctly or if the checks performed on the set $\Xi^{P_1}_{\mathcal{M}_{SM}}(\tau^i_{\Pi^1}, \pi^i)$ fail.

[24] Due to the fixed messages in $\mathcal{M}^{fix}_{SM}$.

The proofs are furthermore collapsing, a simple hybrid argument (similar to the one used in [129] to prove the composability of the collapsing property of commitment schemes, Lemma 16) shows that the advantage of the Adversary is $\epsilon_c + \epsilon_c^{\mathsf{Com}}(M^{fix} + X_\Pi)$, the first factor coming from the collapsing-proof property of the non-compiled CP-maps and the remaining two from the collapsing property of the commitment scheme. Note that here, although the proof contains both messages and opening information, we restrict ourselves to measuring the registers containing messages.

The leakage is also verifiable: if the leak asks for a part of the non-compiled $proof_{sk}$, P_2 can simply use the previous leak verification function $V_{\mathscr{C}}$. Otherwise, the leak corresponds to a message that has been pre-computed by P_1. In that case, the leak verification algorithm verifies that the commitment is valid using Verif and that the position in the message $m_{sk,COM}$ is coherent with the message requested (the sets $\mathcal{S}_M(\tau^i_{\Pi^1}, \pi^i_M)$ are ordered in the leaks according to the round number in which the messages should be requested and each set has a known size).

The sets $\Xi^{P_1}_{\mathcal{M}_{SM}}(\tau^i_{\Pi^1}, \pi^i)$ generated by each call to $S'_{\mathscr{C}'}$ do not leak any information (they can be constructed by P_2 on its own), apart from the messages from the set $\mathcal{M}^{fix}_{SM}$ which have been fixed by P_1. These may each give at most an advantage of ϵ_{pre-c} to the Adversary (this is not necessarily linked to key-indistinguishability, but can be considered as such in the worst case). The combined deterioration in security is therefore $\epsilon_k + \epsilon_{pre-c}$, which remains negligible.

■

We now state the main theorem regarding the properties of the Compiler presented above.

Theorem 4.5 (Compiler Properties)**.** *Let Π be a CC-able Protocol (Definition 4.13) that is ϵ_{cor}-correct, ϵ_V-verifiable for P_1, ϵ_1-secure against quantum polynomial-time $\mathcal{M}_{SM}$-SM adversarial P_1^* (Definition 4.12) and ϵ_2-secure against Malicious P_2^* (Definition 3.7, statistical or computational).*

Let ϵ'_c and ϵ'_k be respectively the proof-collapsing advantage and key-distinguishing advantage of compiled CC-able CP-maps $S'_{\mathscr{C}'}$ and $R'_{\mathscr{C}'}$ and suppose that the commitment scheme $(\mathsf{Com}, \mathsf{Verif})$ used in Transformation 1 is perfectly complete, perfectly-hiding and collapsing. Let $\epsilon_{cc-fail}$ an upper-bound on the failure probability of the input-constraint procedure if using the second condition for CC-Compatible Simulators[25] and ϵ_{SS} the Specious-Simulation distinguishing advantage.

Then the compiled protocol $T(\Pi, s)$ resulting from the application of Transformations 1 and 2 on Π across s executions is:

1. *ϵ_{cor}-correct;*
2. *ϵ_V-verifiable for P_1;*
3. *$s\epsilon_1 + \epsilon_{cc-fail}$-secure against SM P_1^*;*
4. *$s(\epsilon_2 + \epsilon_{SS} + \epsilon'_k) + \epsilon_{cc-fail}$-secure against Malicious P_2^**
5. *$\mathcal{O}(1/\sqrt{s} + s(\epsilon_1 + \epsilon'_c) + \epsilon_{cc-fail})$-secure against Malicious P_1^*.*

Theorem 4.5 is proven as a series of lemmata below. First of all, it is quite easy to see that the transformation preserves correctness.

Lemma 4.6 (Compiler Correctness)**.** *If the initial CC-able Protocol is ϵ_{cor}-correct then so is the transformed version.*

[25] When Π^1 is replaced with an extension $Ext - \Pi^1$ that allows for checking that a player has used the same input in multiple executions of the protocol.

Proof. Suppose that both participants are honest. The transformed protocol is exactly the same as an execution of the original protocol on instance α with additional steps. Since P_1 is honest, it prepares all states, messages and proofs correctly. In protocol Π^1 it sends the correct values for all executions and effectively the parties perform s executions of Π^1 in parallel. After receiving the states and messages and checking the proofs (all of which pass as they have been honestly prepared), the players go on to execute protocol Π^2 on instance α, discarding the rest, with the sole difference being that P_1 uses openings of pre-computed messages instead our the actual values of messages for the leaks. The bit-commitment scheme is perfectly complete and so the commitment verification always succeeds on honestly generated commitments. The execution of Π^2 thus concludes as in the original protocol and the correctness of the compiled version follows (with the same bound). ■

The Compiler also preserves the verifiability of the original protocol for P_1. Note that the same result could be obtained for P_2, but if the original protocol is also verifiable for P_2 then the Compiler is useless as its purpose is precisely to provide a verification procedure for the initial protocol (albeit only for a subset of the protocol's steps).

Lemma 4.7 (Preservation of Verifiability)**.** *If the initial protocol was ϵ_V-verifiable for P_1, then the compiled protocol is also ϵ_V-verifiable for P_1.*

Proof (Sketch) In this proof we consider that the Sender is honest while the Receiver is Malicious. For a protocol to be ϵ_V-verifiable for honest P_1, a fully Malicious P_2^* should not be able to force an incorrect output on P_1 without getting caught with probability higher than ϵ_V. One instance of protocol Π is ϵ_V-verifiable for P_1 and the compiled protocol, in the case of an honest P_1, reduces to a single execution of Π with additional steps (as shown in the proof of correctness above). The commitments that P_2 receives after protocol Π^1 do not reveal any additional information as they are perfectly hiding. The deviation of the P_2 is therefore independent of these commitments. Thus the compiled protocol is also ϵ_V-verifiable for P_1.

■

The security guarantees for both players are also preserved, as expressed by the following two lemmata.

Lemma 4.8 (Preservation of Security against SM P_1^*)**.** *Let Π be a CC-able Protocol that is ϵ_1-secure against $\mathcal{M}_{SM}$-Semi-Malicious P_1^* and let $\epsilon_{cc-fail}$ an upper-bound on the failure of the input-constraint procedure if using the second condition for CC-Compatible Simulators. Then the compiled protocol $T(\Pi, s)$ is $s\epsilon_1 + \epsilon_{cc-fail}$-secure against $\mathcal{M}_{SM}$-SM P_1^*.*

Proof. Let $Sim_{P_1^*}$ be a Simulator against SM adversarial P_1^* in the initial protocol Π. We reuse this Simulator for the initial protocol to construct a Simulator $Sim_{P_1^*}^{T,SM}$ for SM adversarial P_1^* in protocol $T(\Pi, s)$ as described in Simulator 5 below.

Notice that, apart from the Simulator checking the proofs (which pass the test necessarily), the protocol is exactly the same as Π in terms of what an SM-adversarial P_1 is able to achieve, up to the fact that now it can attack s executions of the first protocol (this changes when dealing with fully Malicious Adversaries of course). ■

Simulator 5 Post-Compile Semi-Malicious P_1^*

1. In the first step of the compiled protocol, the Simulator $Sim_{P_1^*}^{T,SM}$ runs one copy of $Sim_{P_1^*}$ for each of the s instances of protocol Π^1. This is done sequentially or concurrently depending on the security of the protocol Π.
2. The simulator fails with probability $\epsilon_{cc-fail}$ if the Adversary is able to use a different input in one of the executions (if the call to the Ideal Functionality is performed during $Ext - \Pi^1$).
3. Otherwise, it chooses and index α, receives the proofs for $i \neq \alpha$ and checks them using $R'_{\mathscr{C}'}$ (all of these checks pass since we are dealing with an SM-Adversary that prepares these messages honestly).
4. It proceeds the same way as $Sim_{P_1^*}$ for Π^2 on execution α until the end of the protocol, at which point it returns whatever $Sim_{P_1^*}$ returns and stops.

Lemma 4.9 (Preservation of Security against P_2^*)**.** *Let Π be an CC-able Protocol that is ϵ_2-secure against Malicious P_2^* for negligible ϵ_2. Suppose that the commitment scheme used in Transformation 1 is perfectly hiding, that the compiled CC-able maps are ϵ'_k-key-indistinguishable and let $\epsilon_{cc-fail}$ be an upper-bound on the failure of the input-constraint procedure if using the second condition for CC-Compatible Simulators and ϵ_{SS} the Specious-Simulation distinguishing advantage. Then the compiled protocol $T(\Pi, s)$ is $s(\epsilon_2 + \epsilon_{SS} + \epsilon'_k) + \epsilon_{cc-fail}$-secure against Malicious P_2^*.*

Proof. Note that is it not possible to use the same strategy as the proof for an adversarial Receiver in the Q-CC Protocol. There, the simulator had to force the Receiver to choose the state sent by the Ideal Resource as the final state and did so by permuting the states and messages until the Receiver picked the correct one (by using the oblivious rewinding technique). Here the state produced by the Simulator may depend on the index of the execution and so this strategy no longer works (i.e. it cannot produce one evaluation state and $s-1$ correct check states and then permute them). However, we can use the fact that it can now control the way that all of these states are produced (while previously it could only control $s-1$ on them). This can be seen as a generalisation of the proof of security of the Q-CC Protocol against Malicious Receiver.

The initial protocol is ϵ_2-secure against a Malicious P_2^*, meaning that there exists a Simulator $Sim_{P_2^*}$ for Malicious Receiver P_2^* in protocol Π such that the views of the (quantum polynomial-time or unbounded depending on the type of security) Distinguisher in the ideal and real world are ϵ_2-close.

Since the Simulator against Malicious P_2^* is both pre-computable and CC-Specious as per the requirements of a CC-able Protocol, there are efficient CPTP maps that correspond to generating the pre-computed sets for the Simulator's messages in $\mathcal{M}_{SM}$ (let $PreC^*$ be such a map) and the "fake" CC-able sets that pass verification and are coherent with the previous transcript of the Simulator (this map is $S^*_{\mathscr{C}}$). Let U_{pre-c} and U_{spe-s} be purifications of these CPTP maps. Note that the CC-Specious property only applies to the initial CC-able sets, not the compiled ones. However, the interaction with the Simulator is indistinguishable from that with an honest party P_1 for adversarial P_2^* and the pre-computed sets are only dependent on the honestly generated CC-able sets and previous transcript. Therefore the CP-map $S^*_{\mathscr{C}}$ can be extended at no cost to also produce pre-computed messages that pass the tests of the compiled Receiver CP-maps $R'_{\mathscr{C}'}$.

Given a Simulator $Sim_{P_2^*}$ for Malicious P_2^* in protocol Π, the Simulator $Sim_{P_2^*}^T$ for adversarial P_2^* in compiled protocol $T(\Pi, s)$ functions as described in Simulator 6 below.

Simulator 6 Post-Compile Malicious P_2^*

1. In the first step of the compiled protocol, the Simulator $Sim_{P_2^*}^T$ runs copies of $Sim_{P_2^*}$ for all s instances of the protocol. These are performed in sequence (or in parallel if protocol Π^1 is concurrently self-composable). If the protocol has been extended to force P_2^* to use the same input in all s executions (because the Simulator for the non-compiled protocol performs the call to the Ideal Functionality during protocol Π^1), the Simulator $Sim_{P_2^*}^T$ uses the input extracted by the first execution of Simulator $Sim_{P_2^*}$ to call the Ideal Functionality $\mathcal{F}$ and sends back the same reply to all Simulators $Sim_{P_2^*}$ when they request the call to the Ideal Functionality. This strategy fails with probability at most $\epsilon_{cc-fail}$.
2. Let $\left\{\left|\psi_S^i\right\rangle\right\}_{i\in[s]}$ the purified state of each uncompiled Simulator $Sim_{P_2^*}$ for P_2^* in the s execution before applying the map generating the CC-able state and message. The Simulator $Sim_{P_2^*}^T$ chooses an index $\hat{\alpha} \in_R [s]$ uniformly at random and applies unitary U_{pre-c} to the state $|\psi_S^{\alpha}\rangle$ and unitary U_{spe-s} to every other one.[26] Let $\left\{\mathcal{X}_S^i, m_{S,COM}^i, proof_{S,OP}^i\right\}_{i\in[s]}$ be the resulting sets ($s-1$ of these are negligibly close to indistinguishable from sets generated honestly and therefore pass the verification procedure while the remaining one is used for the rest of the simulation).
3. It sends the sets $\left\{\mathcal{X}_S^i, m_{S,COM}^i\right\}_{i\in[s]}$ to the Adversary.
4. It receives an index α from the Adversary and if $\alpha = \hat{\alpha}$, its sends $\left\{proof_{S,OP}^i\right\}_{i\neq\alpha}$ to the Adversary. The Adversary checks these sets and all checks pass. Otherwise it uses Watrous' Oblivious Rewinding, chooses another index $\hat{\alpha}$ and repeats until it succeeds. The analysis for this step and the necessary rewinds is the same as in the proof of security of the Q-CC Protocol against Malicious Receiver from Section 4.2.4 and Appendix 4.2.5.1. These steps can be further unitarised by generating a uniform superposition over $\hat{\alpha}$ and applying the unitaries generating the sets in a way that is controlled by this superposition.
5. During execution of protocol Π^2 on instance α, it runs the copy of $Sim_{P_2^*}$ that corresponds to that instance, sending whatever the Simulator would send. It handles the leaks also by sending opening information as in the honest protocol (it then also sends whatever Simulator $Sim_{P_2^*}$ would have sent for the message that were pre-computed). At the end it halts, outputting what $Sim_{P_2^*}$ outputs.

The differences with the initial protocol are that protocol Π^1 is run s times, incurring a cost of $s\epsilon_2$, then there are $s-1$ "fake" sets generated using the CP-map $S_{\mathscr{C}}^*$, at a cost of $s\epsilon_{SS}$, and the Adversary may break the key-indistinguishability of the compiled CP-maps, which happens with probability $s\epsilon_k'$ (the value of which is given in Lemma 4.5). This can be formalised easily using hybrid arguments and first replacing the fake sets with honest executions up to the opening by using the Specious Simulator Definition 4.17 (at a cost of $\epsilon_{SS} + \epsilon_2$ per replaced execution), then upper-bounding the probability that the Adversary is able to distinguish that the honest executions have been performed differently than the simulated one, which can happen either by violating the key-indistinguishability (at a cost $s\epsilon_k'$) or distinguishing the last simulated execution from a real one (a cost of ϵ_2). The compiled protocol is therefore ϵ_2'-secure against an adversarial Receiver, with $\epsilon_2' = s(\epsilon_2 + s\epsilon_{SS} + \epsilon_k') + \epsilon_{cc-fail}$.

■

As further clarification for the different strategy of the Simulator compared to that of Malicious

Receiver in the Q-CC Protocol, we remark that, in Q-CC, the only guarantee is that the state received is correct and there is no link to an execution of a protocol. The Q-CC Protocol as described above is meant to be used as a subroutine in a single protocol while here the Compiler strives to combine the executions of multiple instances of a protocol.

However, in some case it is possible to reuse the Q-CC security statement directly, as is the case in the next lemma which shows that the Compiler boosts the security from the Semi-Malicious case to the fully Malicious one.

Lemma 4.10 (Security Boosting against Malicious P_1^*)**.** *Let* Π *be a CC-able Protocol that is* ϵ_1*-secure against a* $\mathcal{M}_{SM}$*-Semi-Malicious* P_1^* *with* ϵ_c'*-collapsing proofs. Then the compiled protocol* $T(\Pi, s)$ *is* $\mathcal{O}(1/\sqrt{s} + s(\epsilon_1 + \epsilon_c') + \epsilon_{cc-fail})$*-secure against Malicious* P_1^**.*

Proof. This proof is carried out using a hybrid version of the compiled protocol. Using the sequential composability result from Theorem 3.1 (Theorem 3.4 from [66]), the compiled protocol is transformed into one in which both parties have access the Send-Blind-Correct-State f_{send} Ideal Functionality, resulting in Protocol 10. Note that doing so automatically adds a cost in security of $\mathcal{O}(1/\sqrt{s} + s\epsilon_c')$ for P_1^*,[27] where ϵ_c' is the proof-collapsing adversarial advantage from Definition 4.4 for the compiled maps. Note that the problem encountered during the proof of Lemma 4.9 (the impossibility of directly combining multiple executions using the Q-CC procedure and directly reusing the strategy of the Simulator against Malicious Receiver in the Q-CC Protocol) does not occur here since the behaviour of the Simulator against Malicious Sender in the Q-CC Protocol only cares about extracting the proof for the chosen set. This can be done so long as the sets satisfy certain conditions, namely that they are CC-able, which is the case here. How they are created (whether using the honest CP-map or any other mean, in this case an execution of a previous protocol) is of no importance since they stem from a malicious party anyway (as opposed to the proof against malicious Receiver where one set is received from the Ideal Functionality).

Protocol 10 Compiled Hybrid Protocol

1. P_1 and P_2 run the protocol Π^1 s times.
2. P_2 chooses an index $\alpha \in_R [s]$ and sends it to P_2. It then sends dummy input λ to the Ideal Functionality.
3. If honest, P_1 sends inp^α it to the Ideal Functionality. Otherwise, it may send any $(m, proof)$ of its choice.
4. If the Ideal Functionality does not send Abort or Corrupted to both parties (in case of a incorrect message and proof), it sends $(m, info)$ to P_1 and $(\mathcal{X}, m)$ to P_2, where $\mathcal{X}$ is a quantum register.
5. P_1 and P_2 then continue with the execution of protocol Π^2 on the α^{th} instance, with the leaks being handled by calls to the Ideal Functionality.

The initial protocol is ϵ_1-secure against SM P_1^*, meaning that there exists a Simulator $Sim_{P_1^*}$ for Semi-Malicious P_1^* in protocol Π such that the views of the polynomial Distinguisher in the ideal and real world are ϵ_1-close. A Simulator $Sim_{P_1^*}^T$ for Malicious P_1^* in the Compiled Hybrid Protocol 10 works as described in Simulator 7 below.

[27] This is why it can not be used in the proof of Lemma 4.8, as the security bound is no longer negligible.

Simulator 7 Post-Compile Malicious P_1^*

1. Note that none of the messages in $\mathcal{M}_{SM}$ appear in protocol Π^1 and so the SM-Adversary is actually a fully Malicious Adversary in this part of the execution. The Simulator $Sim^T_{P_1^*}$ may therefore run a copy of $Sim_{P_1^*}$ for all s sequential instances of the protocol against Malicious P_1^*. If the protocol has been extended such that the Ideal Functionality may be called by the Simulator during the execution of protocol Π^1 (the second option of Definition 4.16), the Simulator $Sim^T_{P_1^*}$ uses the same strategy as $Sim^T_{P_2^*}$ in the previous proof against Malicious P_2^*: it calls the Ideal Functionality the very first time it is required (as soon as the first instance of Simulator $Sim_{P_1^*}$ asks for it) and reuses the result with the other instances. This strategy fails with probability as most $\epsilon_{cc-fail}$.
2. The Simulator $Sim^T_{P_1^*}$ then chooses an index α and sends it to P_2^*. Because it impersonates the Ideal Functionality f_{send} in the hybrid execution, recovers the pair $(m^\alpha, proof^\alpha)$ sent by the Adversary. It may then run the same steps as the Ideal Functionality by using the extractor $E'_{\mathscr{C}'}$, which extracts the classical description ψ of the state and performs all the other classical checks on the pre-computed messages. If $\psi = \perp$, the Simulator halts and outputs whatever the Adversary does. Otherwise it prepares a quantum register $\mathcal{X}$ with state $|\psi\rangle$.
3. Since all the checks passed it is guaranteed that all pre-computed messages have been prepared honestly, which means that the Malicious Adversary in fact acts as an SM-Adversary in protocol Π^2 and therefore the Simulator $Sim^T_{P_1^*}$ may run the Simulator $Sim_{P_1^*}$ for instance α during the execution of protocol Π^2, forwarding the leaks as the Ideal Functionality would. At the end of the protocol, it returns whatever $Sim_{P_1^*}$ does and halts.

Since s executions of the protocol have been run before the CC round, the resulting security is $s\epsilon_1$. The same-input constraint fails with probability $\epsilon_{cc-fail}$. Replacing the Q-CC Protocol by the ideal f_{send} functionality adds another $1/\sqrt{s} + \mathcal{O}(s\epsilon'_c)$ cost. Combining these costs concludes the proof.

■

To show the power of this Compiler when properly used and the related problems that need to be considered, the next section presents how to apply it to a weakly-secure Two-Party Quantum Computation Protocol based on Measurement-Based Quantum Computing and its derivatives.

4.5 Application to Secure Two-Party Quantum Computation

The Measurement-Based Quantum Computing (MBQC) model is presented first, along with two of its variants: Universal Blind Quantum Computing (UBQC) and Verifiable Blind Quantum Computing (VBQC). Then VBQC is used in a simple but weakly-secure Two-Party Quantum Computation (2PQC) Protocol. Finally the Compiler is applied to this protocol by showing that it satisfies all of the prerequisites, thus automatically boosting its security.

4.5.1 The VBQC-Based 2PQC Protocol

We recall here an abstract description of the VBQC Protocol for a fixed computation $\mathscr{C}$, the full description can be found in Section 3.5.2 as Protocol 5. The Client sends a state $|\psi\rangle$ to the Server, then it sends measurement angles, and receives measurement results in return (these results in turn influence the next measurement angles). Let $\mathbf{r}$ be the auxiliary randomness used by the Client in the

computation of the measurement angles (this is a string corresponding to all parameters $r(v)$, each associated to one vertex v) and let $\mathbf{b}$ be the string of all measurement outcomes given back by the Server. The Client then verifies that the measurements have been performed correctly by means of a deterministic verification function $\texttt{Check}(\cdot)$. Given ψ the classical description of the state, $\mathbf{r}$ and $\mathbf{b}$, it deterministically produces a bit $ok \in \{0,1\}$ indicating whether the computation is accepted or rejected, along with keys for decrypting the output (which is encrypted by the teleportation corrections during the VBQC computation, the keys depending on $\mathbf{r}$ and $\mathbf{b}$).

We now give more details on the structure of the state used for computation $\mathscr{C}$. It can be partitioned into N layers (with N being governed by $\mathscr{C}$). Let I_l be the set of indices of the qubits in the l^{th} layer. Let i_C and i_S be respectively the size of the binary inputs of the Client and the Server respectively. The first layer I_1 corresponds to these inputs and can be further partitioned into $I_A \bigcup_{i\in[i_C]} I_{i,C} \bigcup_{j\in[i_S]} I_{j,S}$ where $I_C := \bigcup_{i\in[i_C]} I_{i,C}$ (respectively $I_S := \bigcup_{j\in[i_S]} I_{j,S}$) represents the space of the Client's input positions (respectively the Server's input positions), and I_A are positions for ancilla qubits. We suppose that the final (output) layer can be similarly partitioned into $I_N = O_C \cup O_S \cup O_A$, where the positions in O_C (respectively O_S) correspond to the Client's (respectively Server's) output locations. Note that in each layer, some of the qubits are used for the computation and others are used as traps (and one trap qubit is associated to each computation qubit).

The angle-update function is given by $\delta(v) = \texttt{Ang} - \texttt{Updt}(\mathscr{C}, \psi, \mathbf{r}, \mathbf{b}, v)$ for $v \in I_1^c \cup I_A$ (non-input qubits and input ancilla qubits) and for inputs $\delta(v) = \texttt{Ang} - \texttt{Updt}(\mathscr{C}, \psi, \mathbf{r}, \mathbf{b}, v) + x_i\pi$ for computation qubits $q \in I_{i,C}$ where $x = x_1||\cdots||x_{i_C}$ is the Client's binary input (and similarly for Server's input y and positions in $I_{j,S}$). This function is public, known to both players.

Protocol 11 presents an abstract version of the VBQC-based 2PQC Protocol that it is fully secure against a Semi-Malicious Sender and a Malicious Receiver, in a hybrid model relying on a trusted third party to compute the OT Ideal Functionality 9.[28]

Instantiating the Bit-Commitment and OT Protocols. The Bit-Commitment scheme and an OT Protocol emulating the OT Ideal Functionality used in this protocol need to fulfil rather strict requirements. The Bit-Commitment needs to be perfectly hiding and collapsing. This may be achieved by using the Unbounded Halevi-Micali Bit-Commitment scheme, using the Merkle-Damgard hash function as a primitive (proven secure in [128]). On the other hand, the OT Protocol should be fully simulatable, statistically secure against the Receiver (Server) and computationally secure against a QPT Sender (Client). The protocol from [112] satisfies these constraints when instantiated with the LWE-based dual-mode encryption and using the messy mode of this encryption scheme, the Common Reference String crs being chosen of size i_S (the size of the Server's input). In this mode and for the chosen encryption scheme, crs can be chosen uniformly at random by using a coin-tossing protocol (therefore no trusted setup is needed).

4.5.1.1 Alternative to Oblivious Transfer in VBQC-Based 2PQC Protocol

It is possible to replace the calls to the OT in the 2PQC Protocol above (for the measurement angles corresponding to the inputs of the Server) by another Ideal Functionality implementing a function that

[28] Recall that this functionality allows one player to receive one of two values from the other player, without gaining any knowledge about the one it did not choose. The player sending the values does not know which value has been chosen.

Protocol 11 Abstract VBQC-Based 2PQC

Inputs: Client has input $x \in \{0,1\}^{i_C}$ and Server has input $y \in \{0,1\}^{i_S}$.
Protocol:

1. The Client chooses uniformly at random the classical description ψ of a VBQC resource-state suitable for computation $\mathscr{C}$ and the associated randomness $\mathbf{r}$ (which only depend on the size of the computation $\mathscr{C}$, i.e. the number of qubits in the base-computation)
2. The Client computes angles $\delta(v)$ for all $v \in I_S$ and both values of y_j using the angle-update function.
3. For all $j \in [i_S]$, the players perform one call to the 1-out-of-2 Oblivious Transfer Ideal Functionality: the Client inputs $(\boldsymbol{\delta}_j^0, \boldsymbol{\delta}_j^1)$ and the Server inputs y_j and receives $\boldsymbol{\delta}_j^{y_j}$, which is the set of measurement angles corresponding to its input for all qubits in set $I_{j,S}$.
4. The Client commits to $\mathbf{r}$ and ψ using a perfectly hiding and collapse-binding commitment Com and sends the commitments along with a quantum register containing $|\psi\rangle$ to the Server.[29]
5. The Server performs an entangling operation $\mathsf{CZ}_{\mathscr{C}}$ (which depends only on the structure of the computation graph, each edge corresponding to a CZ operation between the linked qubits).
6. For each non-output layer $l \in [N-1]$:
   a) For all qubits $v \in I_l$, the Client sends angle $\delta(v)$ computed using function $\mathtt{Ang}-\mathtt{Updt}$ (apart for $v \in I_S$ if $l = 1$, the Server has already received those through the OTs).
   b) The Server performs a measurement described by projectors $\left\{\left|+_{\delta(v)}\right\rangle\!\!\left\langle+_{\delta(v)}\right|, \left|-_{\delta(v)}\right\rangle\!\!\left\langle-_{\delta(v)}\right|\right\}$ on each qubit $v \in I_l$ and sends the measurement results $\mathbf{b}_l \in \{0,1\}^{\#I_l}$ to the Sender. These measurement results are appended to string $\mathbf{b}$ (which is initially the empty string).
7. Upon reaching the last layer N, the Client and the Server perform the following key-release protocol:
   a) The Client sends the indices $\mathbf{v}_t$ to the Server, corresponding to the positions of all traps among the output qubits of the Server, along with measurement angles for these qubits and all the qubits in O_A (the output ancilla qubits).
   b) The Server measures the trap qubits among its output and all ancilla output qubits and sends the measurement results along with the qubits at the Client's output locations O_C, let $\widetilde{\mathcal{O}}_C$ be the corresponding quantum register.
   c) The Client measures the traps among its output qubits in the basis they are supposed to be in and appends the result to $\mathbf{b}$ together with the measurement results received from the Server.
   d) The Client then computes $(ok, k_C, k_S) = \mathtt{Check}(\psi, \mathbf{r}, \mathbf{b})$, if $ok = 1$ the Client sends k_S to the Server along with $\mathbf{v}_c$, which contains the indices of computation qubits among the Server's output O_S (let $\mathcal{O}_S$ be the corresponding quantum register). Otherwise it aborts, outputting Abort and sending it to the Server.
   e) The Client finally decrypts the register $\mathcal{O}_C$ (which corresponds to the sub-register of $\widetilde{\mathcal{O}}_C$ that only contains computation qubits) using the key k_C and sets it as its output.
   f) The Server at the end uses the key k_S supplied by the Client to decrypt register $\mathcal{O}_S$, which corresponds to its output.

[29] We anticipate on the next Section and remark that, although these commitments are not opened during the execution of the protocol, they guarantee the proof-collapsing property of CC-able sets.

is more tailor-made to this specific application. If the measurement angles are transmitted via OT, the bit-length of the transferred messages is 9 for each input bit of the Server. This can be simplified by leveraging the specific structure of these messages. We use first the fact that each measurement angle $\delta \in \Theta$ can be written as $\delta = \delta_2 \frac{\pi}{4} + \delta_1 \frac{\pi}{2} + \delta_0 \pi$ (for $\delta_i \in \{0,1\}$). Since we are only considering classical inputs, let δ^0 and δ^1 be the measurement angles corresponding to a given bit of input of the Server for input values 0 and 1 respectively, for the same qubit in the base-location associated to this input bit. Then $\delta^1 = \delta^0$ for trap and dummy qubits but $\delta^1 = \delta^0 + \pi$ for the computation qubit position, and in that case $\delta_2^1 = \delta_2^0$, $\delta_1^1 = \delta_1^0$ and $\delta_0^1 = \delta_0^0 \oplus 1$. This means that the two most significant bits can be transmitted by the Client without relying on an Ideal Functionality or secure protocol (since these do not change, regardless of the qubit's nature). On the other hand, it cannot directly transmit the value δ_0^0 and have the Server update it using its input because it must only be updated for the computation qubit of each position (and not the trap and dummy qubits).

We can now define the function applied by the new Ideal Functionality that replaces OT. For each input bit of the Server, the Client has as input three bits $(\delta_{0,0}^0, \delta_{0,1}^0, \delta_{0,2}^0)$ corresponding to the least significant bits for the measurement angles of the qubits in the base-location associated to the Server's input (for the input bit value 0), along with an index $c = (c_1, c_0)$ corresponding to the position of the computation qubit in this base-location ($(c_1, c_0) \in \{0,1\}^2$ is the bit representation of $c = 2c_1 + c_0$). On the other hand, the Server has as input its bit-input for this base-location b. We want to impose that the output $\delta_{0,c} = \delta_{0,c}^0 \oplus b$ while the rest is untouched ($\delta_{0,c'} = \delta_{0,c'}^0$ for $c' \neq c$), and that $c \in \{0,1,2\}$ (since there are only three qubits in this base-position). We therefore define the following function $\texttt{Blind} - \texttt{Updt}_\delta : \{0,1\}^6 \to \{0,1\}^4$ that takes as input $(x_0, x_1, x_2, c_0, c_1, b)$ and outputs $(d_0, d_1, d_2, \mathsf{Ok})$ defined by:

$$\texttt{Blind} - \texttt{Updt}_\delta(x_0, x_1, x_2, c_0, c_1, b) = \begin{cases} d_0 = x_0 \oplus b \cdot (c_0 \oplus 1) \cdot (c_1 \oplus 1) \\ d_1 = x_1 \oplus b \cdot (c_0 \oplus 1) \cdot c_1 \\ d_2 = x_2 \oplus b \cdot c_0 \cdot (c_1 \oplus 1) \\ \mathsf{Ok} = c_0 \cdot c_1 \end{cases} \tag{4.38}$$

This simply updates the input value of x_c for the computation position c if the bit b is set to 1. It also allows the recipient to check that the value of c does not overflow via the value Ok. Since we impose that $\mathsf{Ok} = c_0 \cdot c_1 = 0$ (the Server is instructed to reject otherwise), we can simplify the expression above to the form from Equation 4.39. The corresponding circuit is given in Figure 4.3.

$$\texttt{Blind} - \texttt{Updt}_\delta(x_0, x_1, x_2, c_0, c_1, b) = \begin{cases} d_0 = x_0 \oplus b \cdot (c_0 \oplus c_1 \oplus 1) \\ d_1 = x_1 \oplus b \cdot c_1 \\ d_2 = x_2 \oplus b \cdot c_0 \\ \mathsf{Ok} = c_0 \cdot c_1 \end{cases} \tag{4.39}$$

The implementation of this function costs therefore 3 AND operations and 5 XOR gates. Conversely, the bit-OT function is defined as $OT(m_0, m_1, b) = b \cdot m_1 + (1 \oplus b) \cdot m_0 = b \cdot (m_0 \oplus m_1) \oplus m_0$. With the convention that $M = m_0 \oplus m_1$ it can be rewritten as $OT(m_0, M, b) = b \cdot M \oplus m_0$. This has to be done

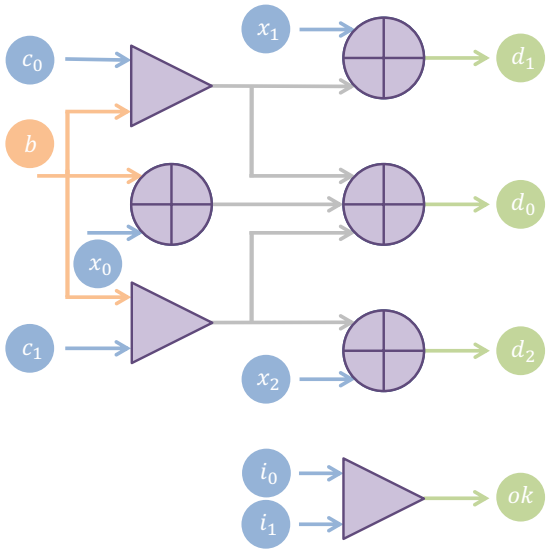


Figure 4.3: Circuit Representation of Alternative Ideal Functionality for Oblivious Input Transmission in QYao

three times as there are three bits to be transmitted and results in a cost of 3 AND gates and 3 XOR gates. Both functions are therefore equivalent in terms of implementation complexity (the higher XOR cost is usually not as important when implementing these functions securely). The added benefit of the new Ideal Functionality is that the values that are transmitted are coherent across different values of the input bit: if using OT, the relation $\delta_1 = \delta_0 + \pi$ for computation qubits is not enforced and so can potentially give more power to an adversarial Sender/Client.

4.5.2 Security Results and Compiler Application

We start by giving the security guarantees of our 2PQC Protocol 11 and then show that it satisfies the constraints of a CC-able Protocol, therefore proving that our Compiler can be applied to it.

4.5.2.1 Security and Verifiability.

We state here the main result regarding the security of Protocol 11.

Theorem 4.6 (Protocol 11 Emulates the 2PQC Ideal Functionality)**.** *Let $\mathcal{M}_{SM} = \{\psi, \boldsymbol{\delta}, \mathbf{v}_t, \mathbf{v}_c, k_S\}$. Suppose that the VBQC Protocol is perfectly correct, perfectly blind for the Server and ϵ_V-verifiable for the Client, that the Bit-Commitment scheme is perfectly-hiding and the OT Protocol is $\epsilon_{OT,S}$-computationally secure against Malicious Sender and $\epsilon_{OT,R}$-statistically secure against Malicious Receiver. Then the 2PQC Protocol 11 is perfectly correct, $i_S\epsilon_{OT,S}$-computationally-secure against $\mathcal{M}_{SM}$-Semi-Malicious Client and $\epsilon_V + i_S\epsilon_{OT,R}$-statistically-secure against unbounded Malicious Server.*

The correctness of Protocol 11 directly follows from that of the VBQC Protocol used in its construction. The only difference in an honest execution is that, instead of the Client sending all measurement angles, some initial angles are chosen by the Server through the OTs. This does not change the correctness.

Also, at the end the Server keeps a subset of the output qubits, which also has no impact on correctness. Therefore it is perfectly correct.

We now prove the security of the base 2PQC Protocol 11 against a Semi-Malicious Client (Lemma 4.11) and a fully Malicious Server (Lemma 4.14). Recall that the hybrid model with OT Ideal Functionality f_{OT} allows the Simulators to impersonate this Ideal Functionality during simulation, i.e. the Simulator receives all inputs that the Adversary sends to the Ideal Functionality and provides the corresponding outputs.

This assumption is used only in order to make the proof clearer and easier, in a real protocol this Ideal Functionality can then be instantiated with an OT Protocol as described in the last paragraph of the previous subsection (using the Sequential Composability of the Stand-Alone Model). This OT Protocol is fully secure against a computationally-bounded Malicious Client, and the security of the resulting 2PQC Protocol is therefore also computational.

The set $\mathcal{M}_{SM}$ of messages that the SM-Adversary must prepare honestly consists of all measurement angles $\boldsymbol{\delta}$, the key k_S and the position of traps and computation qubits $\mathbf{v}_t$ and $\mathbf{v}_c$ among the output qubits of the Server. The security against $\mathcal{M}_{SM}$-SM Client is stated in the following Lemma 4.11.

Lemma 4.11 (Security against Semi-Malicious Client)**.** *Suppose that the OT Protocol is $\epsilon_{OT,S}$-computationally secure against Malicious Sender, then the 2PQC Protocol 11 is $i_S\epsilon_{OT,S}$-computationally-secure against $\mathcal{M}_{SM}$-Semi-Malicious Client.*

Proof. We suppose that the Simulator Sim_{C^*} has single-query access to an oracle $O^{f_{2PQC}}$ which implements the 2PQC Ideal Functionality (described in Ideal Functionality 13 of Section 3.3.3). The proof is performed in the hybrid f_{OT}-model, giving the control of this functionality to the Simulator. Replacing each execution of the OT Protocol by f_{OT} incurs a cost of $\epsilon_{OT,S}$, for a total of $i_S\epsilon_{OT,S}$.

Since the Adversary is Semi-Malicious, the protocol is perfectly equivalent to one where, instead of the Client sending commitments and a state to the Server, it sends instead the commitments (message) and opening information (proof) to an Ideal Functionality f_{send}. The proof is then very straightforward, the Simulator simply recovers $(\psi, \mathbf{r})$ from the Adversary's message to the f_{send} Ideal Functionality (which it impersonates) and uses it to recover the Adversary's input $\tilde{x}$ (by knowing the randomness that was used to generate the angles based on $\tilde{x}$). It sends it to the 2PQC functionality and recovers the Adversary's output register $\mathcal{O}_C$, which it can encrypt as $\widetilde{\mathcal{O}}_C$ (using the correct keys deduced from its knowledge of $(\psi, \mathbf{r})$) to send back to the Adversary at the end. The rest of the protocol is simulated by using a random input $\hat{y}$. The Simulator works as described in Simulator 8 below.

The ideal and real executions in the hybrid model are perfectly indistinguishable for the Environment, therefore the total distinguishing advantage in a non-hybrid execution is $i_S\epsilon_{OT,S}$.

■

While the proof above is almost trivial, it only shows that our Compiler can uplift even protocols which are secure against very weak Adversaries to the full Malicious setting.

We now prove the security against a Malicious Server. To this end, we prove in Lemma 4.12 that the 2PQC Protocol inherits the ϵ_V-local-verifiability for the Client from the VBQC Protocol of [78] directly (stated as Theorem 3.7 in Section 3.5.2).

Lemma 4.12 (Verifiability of 2PQC Protocol)**.** *If the VBQC Protocol is ϵ_V-verifiable for the Client, then Protocol 11 is ϵ_V-verifiable for the Client.*

Simulator 8 2PQC Malicious Client

1. The Simulator impersonates the OT Ideal Functionality and receives $(\boldsymbol{\delta}_j^0, \boldsymbol{\delta}_j^1)$ for all $j \in [i_S]$.
2. It impersonates the f_{send} Ideal Functionality and receives from the Adversary the commitments and opening information, using it to recover $(\psi, \mathbf{r})$. These are honestly prepared as per the definition of an $\mathcal{M}_{SM}$-SM Adversary with $\mathcal{M}_{SM} = \{\psi, \mathbf{r}, \boldsymbol{\delta}, \mathbf{v}_t, \mathbf{v}_c, k_S\}$.
3. When the Adversary sends the input measurement angles for its inputs at layer $l = 1$ (these are also honestly prepared), by knowing the angle-update function $\texttt{Ang} - \texttt{Updt}$, classical description ψ and randomness $\mathbf{r}$, the Simulator can reconstruct the Adversary's bit-input $\hat{x}$: for the angle that the Client sends for its input bit x_i that corresponds to a computation qubit, the Simulator calculates this angle for both values of x_i using the deterministic angle-update function (its knows the other secret parameters) and checks which angle in received (these can always be distinguished since they differ by a value of π).
4. It sends the value $\hat{x}$ to the 2PQC Ideal Functionality and recovers the Adversary's output state in quantum register $\mathcal{O}_C$.
5. The Simulator prepares the state $|\psi\rangle$. It uses a value $\tilde{y}$ chosen uniformly at random and performs the rest of the protocol as though that was the Server's input: it measures the qubits in the correct basis for each value sent by the Adversary and returns the corresponding value. All the traps are measured correctly and the Adversary does not abort at the end based on this information. The measurement angles that the Adversary sends are always honest. As the state is also honestly prepared, the Adversary gains no information from these steps and the ideal and real world are perfectly indistinguishable.
6. For the last layer, it measures the positions corresponding to traps among the Server's output qubits.
7. Using $\psi, \mathbf{r}, \mathbf{b}$ and the function `Check`, it produces a Quantum One-Time-Pad key k_C and uses it to encrypt $\mathcal{O}_C$. It swaps the computation qubits corresponding to the Client's output in the last layer of the graph with the qubits in $\mathcal{O}_C$ and sends the resulting register $\widetilde{\mathcal{O}}_C$ (output and traps) to the Adversary.
8. The Adversary performs the checks on the trap qubits (which pass necessarily), decrypts its output and sends back k_S. At this point the Simulator stops and outputs whatever the Adversary outputs.

Proof. In this proof we consider that the Client is honest while the Server is Malicious. During the first steps of the protocol, the Server only participates in the maliciously secure OT at the end of which it receives its input measurement angles, based on its choice of input y. This step is statistically-secure against Malicious Server and so it is equivalent from its point of view to receiving simply the angles corresponding to its input (it would these angles in the VBQC scheme anyway).

The Server then receives the qubits of all the graphs along with (perfectly hiding) commitments. Deviating at this point on the qubits is equivalent to deviating later. The evaluation part of the protocol follows exactly the same pattern as the VBQC protocol in [78].

We now analyse the key-release step. The Client receives its output qubits and can measure the traps similarly to the VBQC Protocol. For the traps among the Server's output, any deviation on these qubits after the Client has revealed the position of the traps is equivalent to a deviation on the Server's output, which is by definition allowed even in the ideal case (see ρ_{ideal} in Definition 3.12). The Client then verifies that all the traps were measured correctly using the `Check` function.

From this analysis it follows that the exact same verification properties from the protocol in [78] hold for this protocol, namely that our protocol is ϵ_V-verifiable for the Client, which completes the proof.

■

The proof of security against Malicious Server further requires the use of the following Lemma 4.13. It shows that, by possessing a quantum register $\mathcal{O}_S$ containing the output state of the Server, it is possible to produce a state ρ_{Sim} along with a corresponding Quantum One-Time-Pad key k_S and sequence of measurement angles $\delta(v)$, such that the Server receives its correct output after decrypting its output register $\mathcal{O}_S$ with key k_S at the end of the execution with measurement angles $\delta(v)$. This is furthermore indistinguishable from the point of view of the Server from an honest execution, and the state and randomness may be chosen adaptively after setting the value for the measurement angles for the first layer of qubits.

Lemma 4.13 (Deterministic-Output VBQC State)**.** *Let o_S be the size of the Server's output. Given a state ρ_S on o_S qubits, it is possible to efficiently construct a state ρ_{Sim}, measurement angles $\{\delta(v)\}_{v\in O^c}$ for non-output qubits and key k_S such that, at the end of a VBQC evaluation using these measurement angles, the Server's output state is ρ_S. This interaction is furthermore indistinguishable to the Server from an honest execution of the VBQC computation and is verifiable for the Client.*

Proof. We must construct a "fake" graph state whose purpose is to output, for the Server's output qubits in the 2PQC Protocol, a state that is fixed at the creation of this fake graph state (i.e. it has a deterministic output that is known from the start). It must be indistinguishable to the Server from an honestly prepared VBQC state for computation $\mathscr{C}$ (if the secret parameters are not revealed). Suppose that the Client possesses a quantum register $\mathcal{O}_S$ containing the state that it wants to impose as the output of the Server. The graph producing this output after evaluation is constructed as such:

- The Client chooses all parameters of the Dotted Triple-Graph as it would in the normal construction for computation $\mathscr{C}$, apart from the qubits corresponding to the base-locations for the Server's output and the edges linking these positions to the rest of the graph.
- The green vertices of all edges of linking to the Server's output are replaced with red vertices, effectively breaking the output from the rest of the graph. More formally, for each edge base-location e linking to a base-location corresponding to the Server's output, let v_c^e be the index of the corresponding computation (green) qubit in the VBQC graph. The Client chooses a bit $b_{v_c^e} \in_R \{0,1\}$ uniformly at random and sets the state of this qubit as $\left|b_{v_c^e}\right\rangle$. These positions are added to the list of dummy qubits D.
- The Client needs to encrypt the Server's output. For the Server's output base-locations, it chooses a Quantum One-Time-Pad key uniformly at random $(s_X, s_Z) \in_R \{0,1\}^{2\#\mathcal{O}_S}$ and applies the corresponding Pauli operations on quantum register $\mathcal{O}_S$, resulting in state $\mathsf{X}^{s_X}\mathsf{Z}^{s_Z}(\rho_S)$. The qubits of register $\mathcal{O}_S$ are inserted as the computation qubits (green) of these base-locations.
- The decryption keys need to be updated to account for the dummies. Recall that $D_v := N_G(v) \cap D$ is the set of dummy (red) qubits linked to qubit v in the VBQC graph. For $d \in D_v$, let $b_d \in \{0,1\}$ be the value of the dummy. The decryption keys are updated as follows. Let $s_{Z,v}$ be the bit corresponding to the Q-OTP Z-encryption for qubit $v \in O_S$, the updated key is $s'_{Z,v} = s_{Z,v} \bigoplus_{d\in D_v} b_d$. The key s_X remains unchanged and $k_S = (s_X, s'_Z)$. These keys are identical for all computation paths (independent of the values of previous measurements).
- All other parameters are defined as in the normal VBQC construction.

Since the dummies isolate the output of the Server from the rest of the graph, the Server obtains the state $\rho'_S = \mathsf{X}^{s_X}\mathsf{Z}^{s'_Z}(\rho_S)$ in the last layer whatever the previous measurement outcomes are. Thus

applying the updated decryption keys allows it to recover ρ_S. The indistinguishability follows directly from the Server's blindness in the VBQC protocol: since the random parameters are not revealed at any point and only one key is revealed (even though they are identical for all computational branches), the Server cannot get any information from the computation angles that it received. Furthermore, the number and distribution of traps are identical to the ones in the VBQC protocol so the verifiability is preserved for the Client.

■

The security against Malicious Server can then be stated as follows.

Lemma 4.14 (Security against Malicious Server)**.** *Suppose that the VBQC Protocol is perfectly blind for the Server and ϵ_V-verifiable for the Client, that the Bit-Commitment scheme is perfectly-hiding and the OT Protocol is $\epsilon_{OT,R}$-statistically-secure against Malicious Receiver, then the 2PQC Protocol 11 is $\epsilon_V + i_S\epsilon_{OT,R}$-statistically-secure against Malicious Server.*

Proof. The Simulator Sim_{S^*} has single-query access to an oracle $O^{f_{2PQC}}$ which implements the 2PQC Ideal Functionality.

The initial step is to replace each execution of the OT Protocol by the OT Ideal Functionality (incurring a $\epsilon_{OT,R}$ degradation in security for each replacement). The Simulator first recovers the Adversary' input $\hat{y}$ by impersonating this OT Ideal Functionality and calls the 2PQC Ideal Functionality to recover the Adversary's output quantum register $\mathcal{O}_S$. Lemma 4.13 then gives an efficient construction of a state which is indistinguishable to the Server from an honest VBQC state such that it always gives a fixed output if evaluated correctly. The Simulator builds such a state and sends it to the Adversary. Because of the verifiability of the VBQC protocol, any wrong evaluation by the Adversary is detected by the Simulator with high probability, causing it to abort (the probability of aborting on this fake state is the same as in any VBQC honest execution). Therefore if the end of the protocol is reached, the Adversary receives the keys for its output and decrypts it to receive $\mathcal{O}_S$. This is formalised in Simulator 9 below.

This Simulator and proof is the same as the one presented against a Malicious Server in [77]: apart from a slightly different construction for the fake graph, which does not use a deterministic-outcome graph as is the case here but EPR pairs, and different input-injection method. It nevertheless rests on the same principle: the input of the Client is never used and all parameters may be chosen at random. Therefore our protocol inherits the same security analysis and bound.

■

4.5.2.2 Protocol 11 is CC-able.

The following Theorem shows that the previously presented 2PQC Protocol follows the structure of an Abstract Protocol 9 by defining Π^1, Π^2, D_{sk}, input space $\mathbf{I}$, $S_{\mathscr{C}}$, $R_{\mathscr{C}}$, $|\psi_{sk}\rangle$, m_{sk}, $proof_{sk}$, $\mathcal{L}_{\mathscr{C}}$ and $\mathcal{M}_{SM}$. It also satisfies all conditions for the application of the Compiler (Transformations 1 and 2), allowing for its direct application.

Theorem 4.7 (Conditions for CC-able Protocol)**.** *Protocol 11 is a CC-able Protocol:*

- *It is an Abstract Protocol 9 with CC-able CP-maps.*

Simulator 9 2PQC Malicious Client

1. The Simulator invokes the Adversary and receives from it what it would have sent to the trusted party computing the OTs: $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_{i_S})$, which is the actual input that the Malicious Adversary intended to use. It replies with a random $\boldsymbol{\delta}_j^{\hat{y}_j}$ for each call. As mentioned above, the fake state ρ_{Sim} may be chosen later such that these values are correct with regard to that state.
2. It calls the 2PQC Ideal Functionality and receives the Server's output register $\mathcal{O}_S$.
3. It create the fake state ρ_{Sim} and all the associated parameters $\mathbf{r}$ such that they are coherent with the values $\boldsymbol{\delta}_j^{\hat{y}_j}$ sent previously.
4. It sends the fake state to the Adversary, along with commitments to fake messages. The ideal and real situations are again indistinguishable as shown in Lemma 4.13 (intuitively this comes from the blindness property of the VBQC scheme, as here we are simply encoding a computation which always returns the same value contained in $\mathcal{O}_S$).
5. At each round it sends random values $\delta(v)$ and stores the Adversary's replies $b(v)$ (from the construction in the proof of Lemma 4.13 it can be seen that all these values may be chosen at random, apart from the traps which must match the value chosen for state ρ_{Sim}).
6. They perform the same key-swap as in the real execution. It calculates $(ok, k_C, K_S) = \texttt{Check}(\psi_{Sim}, \mathbf{r}, \mathbf{b})$ (where ψ_{Sim} is the classical description of the state ρ_{Sim} up to the Server's output register), if $ok = 1$ the Simulator sends k_S and the output quantum register to the Adversary, after which it stops and outputs whatever the Adversary outputs.

- *It is* pre-computable *for set* $\{\psi, \mathbf{r}, \boldsymbol{\delta}, \mathbf{v}_t, \mathbf{v}_c, k_S\}$ *at round* r^{CC} *(Definition 4.15) for honest Client and the Simulator against Malicious Server;*
- *The Simulators for both players are* CC-Compatible *(Definition 4.16);*
- *The Simulator for Malicious Server is* CC-Specious *(Definition 4.17).*

There is no decrease in security against Malicious Server stemming from revealing one pre-computed and therefore $\epsilon_{pre-c} = 0$*. The Simulator for Malicious Server uses the input-constraint condition of Definition 4.16, but the failure probability is* $\epsilon_{cc-fail} = 0$*. The Specious-Simulation distinguishing advantage is* $\epsilon_{SS} = 0$*.*

Proof. We show the various criteria of a CC-able Protocol in order.
Structure of 2PQC Protocol. The following elements make Protocol 11 an AP:

- The public string $\mathscr{C}$ corresponds to the description of the computation performed by the two parties (expressed as a UBQC graph and uncorrected computation angles, also called measurement pattern), along with the Server's input angles $(\boldsymbol{\delta}_j^0, \boldsymbol{\delta}_j^1)$ for $j \in [i_S]$ sent through the OTs.
- Protocol Π^1 corresponds to the OT Protocols for the Server's input measurement angles (steps 2 and 3 of Protocol 11).
- Step 4 corresponds to round r^{CC}.
- Protocol Π^2 corresponds to the $N-1$ rounds of evaluation (entanglement and measurements on state $|\psi\rangle$), the key-exchange protocol, the computation of $\texttt{Check}(\psi, \mathbf{r}, \mathbf{b})$ by the Client and the final output decryption by the Server (steps 5, 6 and 7).
- D_{sk} returns the empty string (there is no need for a secret key).
- The input space $\mathbf{I}$ is the set of all possible inputs for the Client $\{0,1\}^{i_C}$. It is not used in the state generation here but is important during pre-computations later.

- $|\psi_{sk}\rangle$ is a VBQC resource state $|\psi\rangle$ constructed as described in Section 3.5.2 and VBQC Protocol 5.
- The message m_{sk} contains commitments to ψ and the measurement-hiding parameters $\mathbf{r} \in_R \{0,1\}^{\#\psi}$ chosen uniformly at random, where $\#\psi$ corresponds to the (fixed) number of qubits in a VBQC state for computation $\mathscr{C}$.
- $proof_{sk}$ contains openings for the commitments to the classical description ψ and $\mathbf{r}$.
- $S_{\mathscr{C}}$ produces $|\psi\rangle$ by sampling at random a valid classical description of a VBQC state ψ and producing $|\psi\rangle$ (single qubits, to be entangled later by the Server), then produces m_{sk} and $proof_{sk}$ by sampling $\mathbf{r}$ at random and committing to ψ and $\mathbf{r}$ using Com.
- $R_{\mathscr{C}}$ first calls $E_{\mathscr{C}}$ which receives the openings of commitments, checks that ψ is the description of a valid VBQC state for computation $\mathscr{C}$, checks that the values $(\boldsymbol{\delta}_j^0, \boldsymbol{\delta}_j^1)$ for $j \in [i_S]$ are consistent with the choice of ψ and $\mathbf{r}$ (and correspond to inputs 0 and 1 respectively) and if all checks pass outputs ψ (otherwise it outputs $\perp$). $R_{\mathscr{C}}$ then measures each qubit separately according to the basis specified in the classical description ψ and returns 1 if all measurements succeed (and 0 in any other case).
- There are no valid leaks, meaning that in a given execution neither ψ nor $\mathbf{r}$ should leak to the Adversary.

We can now prove that the defined CP-maps are CC-able:

- An honest Client $S_{\mathscr{C}}$ samples a VBQC state, which is efficient as it simply needs to sample $\#\psi$ rotation angles $\theta(v) \in_R \Theta$ uniformly at random and produce the corresponding states. Sampling the random value $\mathbf{r}$ is also efficient. For all such honestly generated $(|\psi_{sk}\rangle, m_{sk}, proof_{sk})$, the extractor is close to trivial as the classical description of the state is directly included in the proof (along with an opening to a commitment). It only needs to perform a deterministic check on values ψ and $(\boldsymbol{\delta}_j^0, \boldsymbol{\delta}_j^1)$. Then $R_{\mathscr{C}}(|\psi_{sk}\rangle, m_{sk}, proof_{sk}) = 1$ always as it simply measures all qubits in the correct basis if the checks by $E_{\mathscr{C}}$ pass. This is once again efficiently done.
- Due to the collapsing property of the commitment scheme, the proof is collapsing given the message (commitments), hence satisfying Definition 4.4, with the same $\epsilon_c = \epsilon_c^{\mathsf{Com}}$ as the commitment scheme (m_{sk} consists of a single commitment since there is no need to open the individual messages separately as there are no leaks).
- The leakage set is empty so the leaks are trivially verifiable.
- Due to the composable nature of UBQC, as proven in [41] in the Abstract Cryptography framework of [99],[30] leaking all parameters of a given state (before it is used in an execution) does not compromise the blindness of another execution and so the sets are perfectly key-indistinguishable, i.e. with advantage $\epsilon_k = 0$.[31]

The CP-maps verify all the conditions of Definition 4.1 and the structure of the protocol is the same as Protocol 9. The protocol also satisfies all additional conditions for it to undergo the compiling process, as shown below.

[30] This framework is equivalent for two players to the Quantum Universal Composability framework of [126].

[31] Although the property is still called key-indistinguishability, if the key is empty then it is sufficient to prove that revealing the proof of one set does not reveal any information about another one.

Pre-Computability. We start by noting the type of each message. The computation angles and the values sent through the OTs are of type 0, the values of measurement results and the input of the Server are of type 1, the classical description of the state, the values of $\setminus$ and the Client's input are of type 2 and finally the measurement angles sent by the Client, the positions of different qubits among the output of the Server and the keys of the Server's output are of type 3. Note that the only messages that are of type 2 that cannot be fixed (compared to the randomness $\mathbf{r}$) are the input bits of the Client. Therefore the angles associated to the Client's input positions are permuted differently for each execution. The Client needs to pre-compute values $(\boldsymbol{\delta}, k_S)$ since the values of $(\mathbf{v}_t, \mathbf{v}_c)$ are fixed by the state ψ.

The existence of a flow in all the graphs used for computations in MBQC patterns guarantees that the number of dependencies for each qubit does not blow up. The past of qubits $Past(v)$ allows us to calculate upper bounds on the number of dependencies for various computation graphs. More precisely, for the universal brickwork state [21] used here, each qubit has a single X-dependent qubit and at most two Z-dependent qubits, and so the cardinality $\#Past(v)$ is at most 3 for all v. To each influence-past $c(v)$ corresponds a unique value of $\delta(v)$ (the corrected measurement angle for this influence-past), therefore, given $\theta(v)$ (part of ψ) and $r(v)$, the angle that the Client sends to the Server for qubit v is uniquely determined by the result of at most 3 previous measurements, each having value 0 or 1. This results in 8 possible values of $\delta(v)$ in total. Furthermore, the VBQC construction's overhead is linear in the number of qubits in the initial UBQC graph (namely $3\#V + 9\#E$ where $\#V$ is the number of vertices and $\#E$ is the number of edges, with $\#E < 3\#V$ for brickwork states). After applying the VBQC construction on top of a brickwork graph, the number of dependencies for the measurement angles increases but remains constant for each qubit. This is captured by the constant size of the Extended-Past of vertices in the Dotted-Triple Graph (Definition 3.16). Pre-computing the values of $\delta(v)$ is then equivalent in terms of computation for the Client to performing at most a constant number of executions of the VBQC Protocol, which is efficient. The keys needed for decryption of the Q-OTP encrypted output of the Server are similarly dependent only on at most 3 outcomes for each qubit and so are also efficiently pre-computable.

These calculations only use classical values and so have no impact on the correctness of the protocol (the Client does not perform a destructive action on its internal state). This makes the protocol pre-computable for the Sender. The same procedure can be applied by the Simulator for the Server, both satisfying Definition 4.15.

CC-Compatible Simulations. The Simulator for the Client makes the call to the 2PQC Ideal Functionality during sub-protocol Π^2 so it is automatically CC-compatible. However, the call to the Ideal Functionality in the simulation against Malicious Server is made in the part of the protocol which is considered as Π^1. We must therefore constrain the Server to use the same input in all s executions of protocol Π^1 which consists of the call to the OT Ideal Functionality. In the compiled protocol this can be done by simply extending the OT[32] from two strings of length $\#\boldsymbol{\delta}^b_j$ to two strings of length $s\#\boldsymbol{\delta}^b_j$ (each $\delta(v)$ consists of three bits and there are 3 such values in each $\boldsymbol{\delta}^b_j$, for a total of $\#\boldsymbol{\delta}^b_j = 9$ bits). That way the Server receives the angles for all states at the same time for each input, which ensures perfect input-consistency across executions.[33] This satisfies Definition 4.16.

[32] An actual OT implementation replacing the Ideal Functionality must remain secure after such an increase in capacity.
[33] The fake state construction presented in [77] using EPR pairs would allow for performing the call later however.

Server's CC-Specious Simulation. We need to prove that not only the Server's Simulator produces states that pass the checks, but that they are also consistent with the previous transcript. It is simple to see that the Simulator's state are indistinguishable from honestly generated sets since they correspond exactly to correct state for the VBQC computation. However, as mentioned in the proof of security against Malicious Server, the Simulator starts by sending randomly chosen input angles for the Server when it impersonates the OT Ideal Functionality. It needs to do that since it hasn't yet generated the state at this point (the state depends on the reply from the 2PQC Ideal Functionality which is called after the OTs). Nevertheless, the Simulator can choose the state after it sends the values for $\boldsymbol{\delta}_j^b$ in the OTs by simply choosing the values of $\boldsymbol{\theta}$ so that they are coherent with the angles sent to the Server (they correspond to the angles that make the angle-update function $\texttt{Ang}-\texttt{Updt}$ output $\boldsymbol{\delta}_j^b$, and can be computed efficiently). The states generated in this way are indistinguishable from honestly-produced ones. Therefore Definition 4.17 is satisfied.

■

Theorem 4.7 allows us to apply Transformations 1 and 2 to the 2PQC Protocl 11. Using Theorem 4.5, the compiled protocol therefore inherits all the security guarantees given by the lemmata above, namely: correctness, verifiability for the Client (combined with Lemma 4.12), full security against a computationally-bounded Semi-Malicious Client (Lemma 4.11), full security against unbounded Malicious Server (Lemma 4.14). Its security is also boosted and it is thus inverse-polynomially secure against a computationally-bounded Malicious Client. For completeness, we give a description of the compiled protocol in Protocol 12.

4.6 Conclusion and Discussion

We formalised in this chapter two variants of the often-used Cut-and-Choose technique. The classical case for $s-1$ check sets can easily be derived from the quantum presentation by removing the requirements regarding the quantum state and only considering classical messages. Further improving our security bound in the 1-out-of-s scenario beyond the square root factor using techniques based on [26], or conversely finding an attack that saturates this inherently quantum bound, is left as an open question.

Regarding the properties required for the CP-maps to undergo the Q-CC procedure, they are shown to be sufficient but we do not claim that they are necessary and in fact are considering improvements in that regard, hopefully achieving simpler requirements. In that regard, the formalisation of the Fraction Classical Cut-and-Choose Protocol shows an interesting alternative, both in the security framework and requirements for applying the (classical) CC. Our presentation introduces new parameters which various protocols might benefit from, namely the fraction of tests and the maximal fraction of tolerated corruption among the sets transmitted to the Receiver. Then, instead of relying on concentration bounds as most previous result do when analysing this setting, we derive precise and rather tight bounds that are close to being saturated by the optimal attacks. Optimising over the fraction of test runs allows us to tailor this parameter to the tolerated incorrect fraction of a overall protocol and yields, rather counter-intuitively, that the optimal fraction of tests is 3/5 in the case where a majority of honest sets is necessary. Finally, we validate the following intuitive reasoning: using a better error-correcting procedure in the overall protocol such that it can tolerate a larger fraction of incorrect sets will automatically lead

Protocol 12 QYao Cut-and-Choose Protocol

Input: The Client has input $x \in \{0,1\}^{i_C}$ and the Server has input $y \in \{0,1\}^{i_S}$. They have fixed a graph G, flow and measurement angles $\{\phi(v)\}_{v\in V}$ such that the corresponding measurement pattern implements unitary U on graph G.

The protocol:

1. The Client samples s independent copies of the Dotted-Triple Graph computing unitary U (here it just chooses the random parameters for all graphs, it only prepares the actual qubits according to this description later).
2. The Client constructs commitments to all $\boldsymbol{\theta}^i$, $\mathbf{r}^i$, positions of the computation qubits, dummies and traps among the Server's output qubits for all graphs $i \in [s]$.
3. For all graphs $i \in [s]$, the Client precomputes and commits to the values of $\boldsymbol{\delta}^i$ for all possible values of $c^i(v) \in E - Past(v)$ (for non-input vertices), both possible values of $\boldsymbol{\delta}^i$ for input vertices, in permuted order for its own input and in correct order for the Server's input, and all values of the Quantum One-Time-Pad keys for each of the Server's output qubits (also according to flow).
4. The Client and Server participate in i_S instances of a 1-out-of-2 OT Protocol, where in each one the Client's inputs are the sets $\left(\left\{\boldsymbol{\delta}_j^{0,i}\right\}_{i\in[s]}, \left\{\boldsymbol{\delta}_j^{1,i}\right\}_{i\in[s]}\right)$ and the Server's input is bit y_i. After this steps completion, the Server has in its possession the measurement angles corresponding to all their inputs (for the same binary value across all graphs).
5. The Client and the Server perform the Q-CC Protocol:
   a) The Client sends the qubits for all graphs and the commitments.
   b) The Server chooses the evaluation graph index α.
   c) The Client opens the commitments from 2 and 3 for any graph whose index is not α.
   d) The Server performs checks and outputs Abort and halts if any of the checks fail (the checks are the following : the δs are correctly constructed and are compatible with the choice of ϕ, r and θ; the traps and dummies are in the correct place; the decryption keys are correct according to the chosen flow; the values they received for their input via the OTs are consistent with the input bit chosen; they verify that all states are correct by measuring them in the appropriate basis).
6. Then the Client opens the values from commitments to the δs in 3 corresponding to their actual binary input for graph α. The Server entangles the qubits according to the Dotted-Triple Graph and evaluates this graph by asking the Client to open the values to $\boldsymbol{\delta}^\alpha$ in 3 for $c^\alpha(v)$ corresponding to the measurements values they obtained on each qubit. If any of the traps are measured incorrectly the Client privately raises a flag and aborts at the end of the evaluation phase.
7. At the end of the computation they perform the same key-release step as in Protocol 11, but the Client opens the commitments to the positions of the qubits and the values of the Server's keys.

to a better security bound in the FC-CC procedure, thereby decreasing the total number of sets that need to be generated to attain a given security level.

Next, the presented Compiler shows how to use the Q-CC procedure beyond relying solely on the Sequential Composability Theorem 3.1 from Section 3.3.1. Similarly to the Q-CC procedure, we introduce novel properties that allow a protocol to be compiled in order to boost its security. These properties are not proven to be necessary and simplifications may also be possible. Extensions to more varied protocols can be considered as well: for now the parties are constrained by the structure of the AP in the way that they can produce and use the CC-able sets.

Finally, we present a 2PQC Protocol based on the Verifiable Blind Quantum Computation Protocol of [78] and its extension to the two-party case from [77], but for classical inputs and quantum outputs. Instead of relying on Oblivious Transfers, the protocol of [77] leverages a different strategy for inserting inputs in the VBQC graph which allows quantum inputs for the Client and the Server. It is possible to extend our protocol to quantum inputs for the Client by allowing it to teleport its input via an EPR-pair to the correct position for the unchecked graph. This would however require a modification to the Q-CC procedure to allow for quantum proofs, since this new state can only be checked by the Server if it possesses also the half-EPR pair that the Client keeps for its teleportation. This type of mixed-state testing has been investigated in [43], but not in a simulation-based security framework. This modification is already not straight-forward since the proofs must be collapsible for our simulator to function. The problem is altogether different for inserting the Server's input since it must not discover the nature of the qubits amongst its input (computation, dummy or trap). A new blind input-insertion technique would have to be designed while at the same time making sure that the verifiability property of the VBQC protocol is not impacted.

Future Work

New Attack on Classical Yao Protocol by [88]. While developing the framework for FC-CC and in particular after formalising the security requirements (namely the fact that all future decisions should be made based on the secure fraction of evaluation sets), we found an attack that extends the Selective OT Attack described in Section 4.3.3 to a more general Input-Selective Abort Attack. The counter-measure proposed by [88] against the Selective OT Attack is insufficient against this new variant and their protocol is therefore vulnerable to this attack. More precisely, their case analysis for aborts is incomplete. Based on our formalisation of FC-CC, we aim to construct a protocol that is secure against the Input-Selective Abort Attack and prove that Yao's Classical Protocol using FC-CC is composably post-quantum secure under standard assumptions of post-quantum secure symmetric encryption, 1-out-of-2 Oblivious Transfer and Two-Party Coin-Tossing.

Quantum FC-CC and Exponentially-Secure QYao. Another important development that is missing from our results is the extension of the Fraction Cut-and-Choose to the quantum setting. While the protocol and ideal scenario would be fairly similar to the classical setting, problems may arise from entangled attacks across the multiple forwarded states when the outer protocol uses them in later computations. In light of the numerous issues faced by protocols using the FC-CC, defining properly the conditions for applying a compiler to an interactive protocol as in the inverse-polynomial case would

also be a challenge. We give an attack against our QYao protocol using its inherent interactivity which shows concretely why this technique is not applicable in our setting.

In the maliciously-secure classical Yao protocol, the probability of cheating and not getting caught made exponentially small by using the FC-CC technique with $s/2$ check sets and the same number of evaluation sets and revealing only the majority output of the evaluation sets. In our case, a Malicious Client, for one graph and a single trap qubit in the Server's input base-locations, can send through the OTs correctly prepared values of δ if the Server's input is 0 and adds π if the Server's input is 1 (meaning that the measurement outcome is flipped for this trap). If the results of all measurements are given back as is currently the case, Client can recover the Server's input if the corrupt graph is evaluated. This attack succeeds with probability $\frac{\text{number of evaluation graphs}}{s}$ and so we cannot hope for a better security bound than an inverse polynomial with this version of the protocol.

Classically, evaluating an incorrect circuit has minimal influence when using $s/2$ evaluation circuits since only the majority output is returned (although many precautions need to be taken in this case as well). Here guaranteeing that the majority of evaluation circuits is correct would not be enough as the evaluation of a circuit gives extra information for the sake of verifiability. Tricks can be put in place to certify that the traps of a majority of graphs have been computed correctly and that a classical output (most likely deterministic or close to) can be extracted from this majority. This would however require either generic Zero-Knowledge Proofs or a complex Classical Secure Two-Party Computation primitive, something that adds automatically significant overhead to the computation and is not in line with the initial goal of our result.

This would however only solve the question for classical inputs and outputs. Allowing arbitrary quantum inputs is not possible using this strategy since it would mean running multiple instances of the protocol on a single copy of this input. This can be solved by imposing that these quantum have an efficient classical description which is known to the player (for example defined as $|psi\rangle = U|0\rangle$ for efficiently computable U). The player would then generate copies of this input at will for the multiple executions. On the other hand, implementing quantum outputs would require a procedure akin to the final majority vote. Although a method has been presented recently at QIP2021 for quantum majority voting [23] (paper not yet available), it is based on the promise that all input states to the algorithm are either an unknown state $|\phi\rangle$ or orthogonal to $|\phi\rangle$. This would not necessarily be the case here as there is not guarantee that the deviation of the Client would produce states orthogonal to the correct output. For $|+_{\pi/4}\rangle$ states, the Magic State Distillation Procedure is able to produce a magic state of high fidelity from numerous imperfect copies of it. The same principles would need to be generalised to arbitrary states for this technique to work. Finally, a solution could be to perform the multiple executions of the protocol over an input state which has been encoded using a quantum error correcting code and apply the decoding procedure at the end, but this would result in a substantial blow-up in the resource used to implement the protocol.

Covert Adversaries in Quantum Protocols. In the classical setting, a notion called Covert Adversaries has been defined in [11]. It models situations where the Adversary might have a non-negligible chance of cheating in a more elegant way than inverse-polynomial security. If the Simulator has detected a cheating attempt, it sends a signal to the Ideal Functionality who in turn flips a coin with the same bias as the Adversary's probability of cheating in the real world. If the coin-toss is a success, it

forwards the input of the honest party to the Simulator, who can then continue the protocol with the honest player's input. In addition to the problems inherent to this definition,[34] the detection process in the quantum case actually perturbs the checked states and so the Simulator cannot continue the simulation using the honest player's input. If it measures the graph state and finds out it was incorrect, it is supposed to evaluate this graph with the proper input, however, at that point there is not way to reconstruct what the pre-measurement state was. There is no way to undo a quantum measurement and make the Simulator evaluate it correctly as is done in the classical case.

[34]For some protocols, the Simulator may need this honest input before it detects the cheating attempt in case of a successful coin-toss. This imposes a restriction on the class of protocols whose security can be modelled in this framework: at the moment of detection, the simulator must be able to resume an honest execution of the protocol that is consistent both with the (arbitrary) received input and its previous transcript. This is somewhat similar to the CC-Specious Simulation property from Definition 4.17.

Chapter 5

Dispelling Myths on Superposition Attacks: Formal Security Model and Attack Analyses

5.1 Motivation and Overview of Results

5.1.1 Analysis of Existing Security Models

Modelling the security of classical protocols in a quantum world (especially multi-party protocols) is tricky, since various arbitrages need to be made concerning the (quantum or classical) access to channels and primitives.

A first possibility is to consider classical protocols embedded as quantum protocols, thus allowing the existence of superposition attacks. However, in such a setting, previous results only consider *perfect security*, meaning that the messages received by each player do not contain more information than its input and output. The seminal papers starting this line of work are those proving the impossibility of bit commitment [94, 100]. The perfect security of the protocol implies that no additional information is stored in the auxiliary quantum registers of both parties at the end of the protocol and can therefore be traced out, so that an Adversary can easily produce a superposition of inputs and outputs.

This is for example the approach of [33], and [122], where the perfect correctness requirement is in fact a perfect (unconditional) security requirement (the protocol implements the functionality and *only* the functionality). In [33], they consider an even more powerful adversarial scenario where not only the honest player's actions are described as unitaries (their inputs are also in superposition) but the Adversary can corrupt parties in superposition (the corruption is modelled as an oracle call whose input is a subset of parties and which outputs the view of the corresponding parties). Both papers show that protocols are insecure in such a setting: In [33], they show that in the case of a multi-party protocol implementing a general functionality (capable of computing any function), no Simulator can perfectly replicate the superposition of views of the parties returned by the corruption oracle by using only an oracle call to an Ideal Functionality. In the case of a deterministic functionality, they give a

necessary and sufficient condition for such a Simulator to exist, but which cannot be efficiently verified and is not constructive. In [122], they prove that any non-trivial Ideal Functionalities that accept superposition queries (or, equivalently, perfectly-secure protocols emulating them) must leak information to the Adversary beyond what the classical functionality does (meaning that the Adversary can do better than simply measure in the computational basis the state that it receives from the superposition oracle). In both cases, they heavily rely on the assumption of unconditional security to prove strong impossibility results and their proof techniques cannot be applied to the computational setting.

The second possibility to model the security of classical protocols in a quantum world is to define security models with purely classical messages, in the sense that all supposedly classical messages are measured in the computational basis upon reception. This is the path taken by the Stand-Alone Model of [66] or the Quantum UC Model of [126], which is equivalent in the two-party case to the AC Framework presented in Section 3.3.2. Some (computationally) secure protocols exist in this setting, as shown by a series of articles in the literature (e.g. [91]). However, these models forbid by construction the analysis of superposition attacks, precisely since all classical communications are modelled as measurements.

The Missing Link. The results of [122, 33] in the unconditional security setting are not directly applicable to a *Computationally-Bounded Adversary*. The premiss to their analyses is that since the perfect execution of non-trivial functionalities is insecure, any real protocol implementing these functionalities is also insecure against Adversaries with quantum access (even more since they are simply computationally secure). However it turns out that, precisely because the protocol is only computationally-secure, the working registers of the parties cannot be devoid of information as is the case in the perfectly-secure setting (the messages contain exactly the same information as the secret inputs of the parties, but it is hidden to computationally-bounded Adversaries) and the techniques used for proving the insecurity of protocols in the perfect scenario no longer work.

This issue has been partially solved for single-party protocols with oracle queries in the line of work from [17], but never extended fully to the multi-party setting. The difficulty arises by the interactive property of such protocols. Indeed, in a real protocol, more care needs to be taken in considering all the registers that both parties deal with during the execution (auxiliary qubits that can be entangled due to the interactive nature of the protocols). Furthermore, care must also be taken in how the various classical operations are modelled quantumly, as choosing standard or minimal oracle representations may influence the applicability of some attacks [74]. The naive implementation of superposition attacks, applied to a real-world protocol, often leads to a joint state of the form $\sum_{x,m_1,m_2} |x\rangle |m_1\rangle |m_2\rangle |f(x,y)\rangle$ for a given value y of the honest player's input, and with the second register (containing the set of messages m_1 sent by the Adversary) being in the hands of the honest player (m_2 is the set of messages sent by the honest player and $f(x,y)$ is the result for input x). This global state does not allow the known attacks (such as [70]) to go through as the message registers cannot simply be discarded. This shows that the simple analysis of basic ideal primitives in the superposition attack setting is not sufficient to conclude on the security of the overall computationally-secure protocol and motivates the search for a framework for proving security of protocols against such attacks.

On the Importance of Superposition Attack Analysis. The reader might wonder why it may be important to consider an attack which can be mitigated by simply measuring the incoming supposedly

classical messages or simply not implement classical primitives on quantum hardware. This second point is already moot since there has been recent active research regarding the improvement of implementations of cryptographic primitives on quantum computers, such as [136] which reduces by 30% the number of required qubits for implementing AES encryption and decryption (which is precisely the scheme considered in this work, see Section 5.3.1.2). Also, while the fact that a full computational basis measurement nullifies any superposition attack is correct, this means that an additional security assumption is needed, namely that the quantum device is trusted when required to perform the measurement. This is a common assumption and an Adversary with inside access to the laboratory can potentially perform more devastating attacks. We argue however that this adds a requirement for securing yet another system against outside access in a world where even air-gapped machines have been shown to be breachable if the stakes are high enough (e.g. Stuxnet attack). If the quantum device is connected to a quantum external channel, even protocols that were proven unconditionally-secure (Quantum Key Distribution) have shown vulnerabilities to side channel attacks such as detector blinding, an example of the techniques called *quantum hacking* (see [56] and related works). On the other hand, proving resistance to superposition attacks automatically removes these possibilities without further resource expenditure and it can therefore be seen as closely related to the questions arising in the field of Device-Independent Quantum Cryptography (where no trust is placed in the devices performing the protocol).

5.1.2 Our Contribution

The main purpose of our work is thus to bridge a gap between the following two settings:

- Superposition attacks are allowed and can be analysed, but either in a perfectly-secure setting [33, 122] (both works preclude the existence of secure protocols by being too restrictive) or only for single-party primitives with oracle access [17].
- Computational security of multi-party protocols is defined in [126, 66] but superposition attacks are explicitly forbidden as classical messages are measured.

To our knowledge, our result is the first attempt to formalise a security notion capturing security of two-party protocols against superposition attacks with computationally-bounded Adversaries as a simulation-based definition. We consider a more realistic scenario where a computational Adversary corrupts a fixed set of players at the beginning of the protocol and the input of the honest players are fixed classical values. We suppose that the ideal world trusted third party always measures its queries (it acts similarly to a classical participant), while the honest player always performs actions in superposition unless specifically instructed by the quantum embedding of the protocol (the Adversary and the Simulator can do whatever they want). Security is then defined by considering that an attack is successful if an Adversary is able to distinguish between the real and ideal executions with non-vanishing probability. The reason for adding a measurement to the functionality is to enforce that the (supposedly classical) protocol behaves indeed as a classical functionality. This is further motivated by the results of previous papers proving that functionalities with quantum behaviour are inherently broken.

Case Studies. We show that our proposed security model is satisfiable by proving the superposition-resistance of the classical One-Time-Pad protocol for implementing a Confidential Channel. Conversely, we also present an attack on a slight variant of the Honest-but-Curious version of the classical Yao's

protocol [135] for Secure Two-Party Computation. On the other hand, it is secure against Adversaries that have a quantum computer internally but send classical messages, therefore showing a separation. The variant is presented to demonstrate unusual and counter-intuitive reasons for which protocols may be insecure against superposition attacks.

Proof Technique. During the superposition attack, the Adversary essentially makes the honest player implement the oracle call in Deutsch-Jozsa's (DJ) algorithm [37] through its actions on a superposition provided by the Adversary. The binary function for which this oracle query is performed is linked to two possible outputs of the protocol. The Adversary can then apply the rest of the DJ algorithm to decide the nature of the function,[1] which allows it to extract the XOR of the two outputs. Similarly to the DJ algorithm where the state containing the output of the oracle remains in the $|-\rangle$ state during the rest of the algorithm (it is not acted upon by the gates applied after the oracle call), the Adversary's actions during the rest of the attack do not affect the output register. Interestingly, this means that the attack can thus also be performed on the same protocol but where the Adversary has no output.

Superposition-Secure Two-Party Computation. Counter-intuitively, it is therefore not the output that makes the attack possible, but in this case the attack vector is a message consisting of information that, classically, the Adversary should already have, along with a partial measurement on the part of the honest player (which is even stranger considering that it is usually thought that the easiest way to prevent superposition attack is to measure the state). This shows that adding extra communication, even an exchange of classical information which seems meaningless for classical security, can make the protocol become subject to superposition attacks. Removing the point of failure by never sending back this information to the Adversary (as is the case in the original Yao Protocol) makes the protocol very similar in structure to the One-Time-Pad Protocol, where one party sends everything to the other, who then simply applies local operations. The proof for the One-Time-Pad works by showing that there is a violation of the no-signalling condition of quantum mechanics if the Adversary is able to distinguish between ideal and real scenarios. In fact, if it were able to gain any information, it would be solely from the local operations performed by the honest player, which would imply that information has been transferred faster than the speed of light. This technique can only be reused if the honest party in Yao's protocol does not give away the result of the measurement on its state (by hiding the fact that it either succeeded in completing the protocol or aborted if it is unable to do so correctly). We show that Yao's Protocol is secure against superposition attacks if the (honest) Evaluator recovers the output and does not divulge whether or not it has aborted.

Contribution Summary and Outline.

- Section 5.2 gives a new security model for superposition attacks;
- Section 5.3 describes a variant of Yao's Protocol and proves its security against adversaries exchanging classical messages;
- Section 5.4.2 demonstrates a superposition attack against this modified Yao's Protocol. This attack is then applied in Section 5.4.5 to an Oblivious Transfer protocol with slightly improved attack success probability;

[1] Recall that the DJ algorithm decides whether a binary function is balanced or constant

- Section 5.5 proves the superposition-resistance of two protocols. First we show that the Classical One-Time-Pad Protocol remains secure in our framework. We then leverage the knowledge acquired through our attack in the previous Section to build a secure version of Yao's Protocol.

5.2 New Security Model for Superposition Attacks

We start by presenting a new model for security against superposition attacks. All protocols in this chapter will again be two-party protocols (between parties P_1 and P_2). P_1 will be considered as the Adversary (written P_1^* when corrupted), while P_2 is honest. Although we consider purely classical protocol, in order to be able to execute superposition attacks, both parties will have access to multiple quantum registers, respectively denoted collectively $\mathcal{X}$ and $\mathcal{Y}$. All communications are considered as quantum unless specified.

The principle of superposition attacks is to consider that a player, otherwise honestly behaving, performs all of its operations on quantum states rather than on classical states. In fact, any classical operation defined as a binary circuit with bit-strings as inputs can be transformed into a unitary operation that has the same effect on each bit-string (now considered a basis state in the computational basis) as the original operation by using Toffoli gates (as explained in Section 2.2.2.2). Although any quantum computation can be turned into a unitary operation (using a large enough ancillary quantum register to purify it, as seen in Section 2.2.2), it may be that the honest player may have to take a decision based on the value of its internal computations. This is more naturally defined as a measurement, and therefore such operations will be allowed but only when required by the protocol (in particular, when the protocol branches out depending on the result of some computation being correct). The rest of the protocol (in the honest case) will be modelled as unitary operations on the quantum registers of the players.

General Protocol Model. We assume that the input of the honest player is classical, meaning it is a pure state in the computational basis, unentangled from the rest of the input state (which corresponds to the Adversary's input). This is in stark contrast with other papers considering superposition attacks [122, 33] where the inputs of the players are always a uniform superposition over all possible inputs. We also consider that the corrupted party is chosen and fixed from the beginning of the protocol. We will often abuse notation and consider the corrupted party and the Adversary as one entity.

The security of protocols will be defined using the *real/ideal simulation paradigm*, adapted from the Stand-Alone Model of [66]. The parties involved are: an Environment $\mathcal{Z}$, the parties participating in the protocol, a Real-World Adversary $\mathcal{A}$ and an Ideal-World Adversary also called Simulator $\mathcal{S}$ that runs $\mathcal{A}$ internally and interacts with an Ideal Functionality (that the protocol strives to emulates). An execution of the protocol (in the real or ideal case) works as follows:

1. The Environment $\mathcal{Z}$ produces the input y of P_2, the auxiliary input state $\rho_{\mathcal{A}}$ of the Adversary (containing an input for corrupted party P_1^*, possibly in superposition).
2. The Adversary interacts with either the honest player performing the protocol or a Simulator with single-query access to an Ideal Functionality.

3. Based on its internal state, it outputs a bit corresponding to its guess about whether the execution was real or ideal. If secure, no Adversary should be able to distinguish with high probability the two scenarios.
4. The Adversary sends a state to the Environment $\mathcal{Z}$.
5. The Environment $\mathcal{Z}$ takes as input this final state and outputs a bit corresponding to its guess of whether the execution was real or ideal.

Network Model. To capture both the security against Adversaries with and without superposition (so that we may compare both types of security for a given protocol), we parametrise the security Definition 5.1 below with a network model $\mathfrak{N}$. The quantum network $\mathfrak{Q}$ is modelled by having both players interact not only with their internal quantum registers but also with a shared quantum communication register $\mathcal{Q}$. These actions are defined as unitaries. On the other hand, the classical network $\mathfrak{C}$ is modelled as both players having access to a shared classical tape $\mathcal{C}$ which is read the beginning of each activation of a player and a quantum register initialised using the computational basis vector corresponding to the message contained within (or equivalently, the shared quantum register $\mathcal{Q}$ from the quantum network is measured in the computational basis). The outgoing messages are written to the tape at the end of each player's activation. The case where the network is classical is called *classical-style* security (as it is simply a weaker variant of Stand-Alone Security in the usual sense of [66]), while a protocol that remains secure when the network is quantum is said to be *superposition-resistant*. This allows us to demonstrate a separation between Adversaries with and without superposition access. Furthermore, since the classical network can be seen as a restricted quantum channel (any measurement can only destroy information, therefore the Adversary is strictly less powerful), security with superposition access automatically implies classical-style security.

Ideal Functionality Behaviour and Formal Security Definition. This section differs crucially from previous models of security. The Classical Two-Party Computation Ideal Functionality implementing a binary function f, formally defined as Ideal Functionality 19, takes as input a quantum state from each party, measures it in the computational basis, applies the function f to the classical measurement results and returns the classical inputs to each party while one of them also receives the output. [2]

While it can seem highly counter-intuitive to consider an ideal scenario where a measurement is performed (since it is not present in the real scenario), this measurement by the Ideal Functionality is necessary in order to have a meaningful definition of security. It is only if the protocol with superposition access behaves similarly to a classical protocol that it can be considered as resistant to superposition attacks. It is therefore precisely because we wish to capture the security against superposition attack, that we define the Ideal Functionality as purely classical (hence the measurement). If the Ideal Adversary (a Simulator interacting classically with the Ideal Functionality) and the Real Adversary (which can interact in superposition with the honest player) are indistinguishable to the Environment, only then is the protocol superposition-secure.

[2] In the classical case, it is argued in [89] that it suffices without loss of generality to describe the ideal functionality for functions where only one party receives an output, in this case P_1, via the following transformation: any function inputs (x, y) and two outputs (w, z) to two parties can be transformed into a function taking as input $((x, p, a, b), y)$ and outputting to a single party $(w, \alpha := z \oplus p, \beta := a \odot \alpha \oplus b)$, where $\oplus$ and $\odot$ are the addition and multiplication operations in a well-chosen finite field (p serves as a perfect One-Time-Pad of the output z and β serves as a perfect One-Time Message Authentication Code of α). As shown in Section 5.5.2 this is not so clear in our model.

Ideal Functionality 19 Two-Party Secure Function Evaluation

Public information: Binary function $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \longrightarrow \{0,1\}^{n_Z}$ to be computed (where n_X, respectively n_Y, is the size of the input of P_1, respectively P_2, and n_Z is the size of the output).
Inputs: P_1 has classical input $x \in \{0,1\}^{n_X}$ and P_2 has classical input $y \in \{0,1\}^{n_Y}$.
Computation by the Functionality:

1. If the trusted party receives an input which is inconsistent with the required format (different input size) or Abort, it sends Abort to both parties. Otherwise, let $\tilde{\rho}_{in}$ be the input state it received from P_1 and P_2.
2. The trusted party measures the parts of $\tilde{\rho}_{in}$ in registers $\mathcal{X}$ and $\mathcal{Y}$ in the computational basis, let $(\tilde{x}, \tilde{y})$ be the outcomes of the measurement.
3. The trusted party computes $\tilde{z} = f(\tilde{x}, \tilde{y})$ and sends $(\tilde{x}, \tilde{z})$ to P_1 and $\tilde{y}$ to P_2.

Furthermore, as argued briefly above, Ideal Functionalities which do not measure the inputs of both parties when they receive them as they always allow superposition attacks, which then extract more information than the classical case (as proven in [122]). A superposition attack against a protocol implementing such a functionality is therefore not considered an attack since it is by definition a *tolerated behaviour in the ideal scenario.*

We can now give our security Definition 5.1. A protocol between parties P_1 and P_2 is said to securely compute two-party functions of a given set $\mathfrak{F}$ against corrupted party P_1^* if, for all functions $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \longrightarrow \{0,1\}^{n_Z}$ with $f \in \mathfrak{F}$, no Environment $\mathcal{Z}$ can distinguish between the real and ideal executions with high probability.

Definition 5.1 (Computational Security in Network Class $\mathfrak{N}$)**.** *Let $\epsilon(\eta) = o(1)$ be a function of the security parameter η. Let $f \in \mathfrak{F}$ be the function to be computed by protocol Π between parties P_1 and P_2. We say that a protocol Π $\epsilon(\eta)$-securely emulates Ideal Functionality $\mathcal{F}$ computing functions from set $\mathfrak{F}$ against adversarial P_1^* in network $\mathfrak{N}$ (with $\mathfrak{N} \in \{\mathfrak{C}, \mathfrak{Q}\}$) if for all quantum polynomial-time Adversaries $\mathcal{A}$ controlling the corrupted party P_1^* and Environments $\mathcal{Z}$ producing y and $\rho_{\mathcal{A}}$, there exists a Simulator $S_{P_1^*}$ such that, in network $\mathfrak{N}$:*

$$\left|\mathbb{P}\Big[b = 0 \mid b \leftarrow \mathcal{Z}\Big(v_{\mathcal{A}}(S_{P_1^*}, \rho_{\mathcal{A}})\Big)\Big] - \mathbb{P}\Big[b = 0 \mid b \leftarrow \mathcal{Z}\Big(v_{\mathcal{A}}(P_2(y), \rho_{\mathcal{A}})\Big)\Big]\right| \leq \epsilon(\eta) \tag{5.1}$$

In the equation above, the variable $v_{\mathcal{A}}(S_{P_1^*}, \rho_{\mathcal{A}})$ corresponds to the final state (or view) of the Adversary in the ideal execution when interacting with Simulator $S_{P_1^*}$ with Ideal Functionality $\mathcal{F}$ and $v_{\mathcal{A}}(P_2(y), \rho_{\mathcal{A}})$ corresponds to the final state of the Adversary when interacting with honest party P_2 in the real protocol Π. The probability is taken over all executions of protocol Π.

In the case where one party does not receive an output, it is possible to reduce the security property to input-indistinguishability, defined below in Definition 5.2.

Definition 5.2 (Input-Indistinguishability in Network Class $\mathfrak{N}$)**.** *Let Π be protocol between parties P_1 and P_2 with input space $\{0,1\}^{n_Y}$ for P_2. We say that the execution of Π is* ϵ-input-indistinguishable *for P_1^* in network $\mathfrak{N}$ if there exists an $\epsilon(\eta) = o(1)$ such that, for all computationally-bounded quantum Distinguishers $\mathcal{D}$ and any two inputs $y_1, y_2 \in \{0,1\}^{n_Y}$:*

$$\left|\mathbb{P}\Big[b = 0 \mid b \leftarrow \mathcal{D}\Big(v_{\mathcal{A}}(P_2(y_1), \rho_{\mathcal{A}})\Big)\Big] - \mathbb{P}\Big[b = 0 \mid b \leftarrow \mathcal{D}\Big(v_{\mathcal{A}}(P_2(y_2), \rho_{\mathcal{A}})\Big)\Big]\right| \leq \epsilon(\eta) \tag{5.2}$$

In the equation above, the variable $v_{\mathcal{A}}(P_2(y_i), \rho_{\mathcal{A}})$ corresponds to the final state of the Adversary when interacting with honest party P_2 (with input y_i) in the real protocol Π. The probability is taken over all executions of protocol Π.

We can now state Lemma 5.1, showing the equivalence of our security notions in the case where the attacker has no output.

Lemma 5.1 (Input-Indistinguishability to Security)**.** *Let $f \in \mathfrak{F}$ be the function to be computed by protocol Π between parties P_1 and P_2, where $\mathfrak{F}$ is the set of functions taking as input $(x, y) \in \{0,1\}^{n_Y} \times \{0,1\}^{n_X}$ and outputting $z \in \{0,1\}^{n_Z}$ to P_2 (and no output to P_1). If the protocol is input-indistinguishable for adversarial P_1^* in network $\mathfrak{N}$ (Definition 5.2) then it is secure against adversarial P_1^* in network $\mathfrak{N}$ (Definition 5.1) with identical bounds.*

Proof. If we suppose that the protocol is input-indistinguishable for Adversaries in network $\mathfrak{N}$, then no computationally-bounded quantum Distinguisher (represented as a an efficient quantum machine acting on the state returned by the Adversary) can distinguish between an execution with inputs y_1 and y_2. The Simulator against an Adversary in network $\mathfrak{N}$ then simply runs the protocol honestly with a random input $\tilde{y}$ (it does not need to call the Ideal Functionality as the adversarial player has no output). Therefore:

$$\left|\mathbb{P}\Big[b = 0 \mid b \leftarrow \mathcal{D}\Big(v_{\mathcal{A}}(P_2(y), \rho_{\mathcal{A}})\Big)\Big] - \mathbb{P}\Big[b = 0 \mid b \leftarrow \mathcal{D}\Big(v_{\mathcal{A}}(S_{P_1^*}(\tilde{y}), \rho_{\mathcal{A}})\Big)\Big]\right| \leq \epsilon(\eta) \tag{5.3}$$

Since this is the case for any efficient distinguisher, it also means that the probability that the Environment outputs a given bit as the guess for the real or ideal execution is the same up to ϵ in both cases. Therefore the protocol is secure.

■

Adversarial Classes. Quantifying Definition 5.1 and 5.2 over a subset of Adversaries in each class yields flavours such as Honest-but-Curious or Malicious. The behaviour of an Honest-but-Curious Adversary in a classical network $\mathfrak{C}$ is the same as a classical Honest-but-Curious Adversary during the protocol but it may use its quantum capabilities in the post-processing phase of its attack. We define an extension of these Adversaries in Definition 5.3: they are almost Honest-but-Curious in that there is an Honest-but-Curious Adversary whose Simulator also satisfies the security definition for the initial Adversary. This is required as the adversarial behaviour of our attack presented later is not strictly Honest-but-Curious when translated to classical messages, but it does follow this new definition.

Definition 5.3 (Extended Classical Honest-but-Curious Adversaries)**.** *Let Π be a protocol that is secure according to Definition 5.1 against Honest-but-Curious Adversaries in a classical network $\mathfrak{C}$. We say that an Adversary $\mathcal{A}$ is Extended Honest-but-Curious if there exists an Honest-but-Curious Adversary $\mathcal{A}'$ such that the associated Simulator $\mathcal{S}'$ satisfies Definition 5.1 for $\mathcal{A}$ if we allow it to output* Abort *when the honest party would abort as well.*

Comments on the Security Model. In our security model, both the Adversary and Simulator can have superpositions of states as input. However, if the Simulator chooses to send a state to the Ideal Functionality, it knows that this third party will perform a measurement on it in the computational basis. Note that in any security proof, the Simulator may choose not to perform the call to the Ideal Functionality. This is because the security definition does not force the Simulator to reproduce faithfully the output of the honest Client, as the distinguishing done by the Environment takes only the Adversary's output into account. This also means that sequential composability explicitly does not hold with such a definition, even with the most basic functionalities (whereas the Stand-Alone Framework of [66] guarantees it). An interesting research direction would be to find a composable framework for proving security against superposition attacks and we leave this as an open question. The subtlety of our attack vector presented below tends to suggest a negative answer.

5.3 The Modified Honest-but-Curious Yao Protocol

In order to demonstrate the capabilities of our new model in the case of more complex two-party scenarios, we will analyse the security of Yao's Protocol in classical and quantum networks.

We start by presenting definitions for symmetric encryption schemes in Section 5.3.1.1. Then in Section 5.3.2 we give a description of a slight variant of the original Yao Protocol, resulting in the Modified Yao Protocol. We show that the modifications do not make the protocol less secure in classical networks, but will make superposition attacks possible as presented in Section 5.4.

5.3.1 Security and Superposition-Compatibility of Symmetric Encryption

5.3.1.1 Definitions For Symmetric Encryption Schemes

We give here the definitions of the properties required from the symmetric encryption scheme to guarantee the correctness and classical-style security of Protocol 13 (given on page 136). Recall that such a scheme is defined as two classical efficiently computable deterministic functions $(\mathsf{Enc}, \mathsf{Dec})$ over $\mathfrak{K}$, $\mathfrak{A}$, $\mathfrak{M}$ and $\mathfrak{C}$ (respectively the key, auxiliary input, plaintext and ciphertext spaces). For simplicity we will suppose that the key-generation algorithm simply samples the key uniformly at random from the set of valid keys.

The symmetric encryption scheme uesd here must satisfy slightly different properties compared to the original protocol of [135] or [89]. The purpose of these modifications is to make it possible to later represent the action of the honest player (the decryption of garbled values) using a Minimal Oracle Representation (see section 2.2.2.2) when embedded as a quantum protocol. We give sufficient conditions implying this definition (as shown in Lemma 5.2) and a concrete instantiation of a symmetric encryption scheme that satisfies them.

Definition 5.4 (Minimal Oracle for Symmetric Encryption)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme defined as above, we say that it has a Minimal Oracle Representation if there exists efficiently computable unitaries* $\mathsf{M}_{\mathsf{Enc}}$ *and* $\mathsf{M}_{\mathsf{Dec}}$*, called minimal oracles, such that for all* $k \in \mathfrak{K}$*,* $\mathrm{aux} \in \mathfrak{A}$ *and* $m \in \mathfrak{M}$*,* $\mathsf{M}_{\mathsf{Enc}} |k\rangle |\mathrm{aux}\rangle |m\rangle = |e_K(k)\rangle |e_A(\mathrm{aux})\rangle |\mathsf{Enc}_k(\mathrm{aux}, m)\rangle$ *(in which case* $\mathsf{M}_{\mathsf{Enc}}^{\dagger} = \mathsf{M}_{\mathsf{Dec}}$*), where* e_K *and* e_A *are efficiently invertible permutations of the key and auxiliary value.*

We now give sufficient conditions on the symmetric encryption scheme so that it may be efficiently represented as a minimal oracle. The first of those is that the encryption and decryption functions are perfect inverses of each other.

Definition 5.5 (Encryption Correctness)**.** *An encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *as defined above is said to be* correct *if, for all keys* $k \in \mathfrak{K}$ *and auxiliary input* $\mathrm{aux} \in \mathfrak{A}$*,* $\mathsf{Dec}_k(\mathrm{aux}, \cdot) \circ \mathsf{Enc}_k(\mathrm{aux}, \cdot) = \mathsf{Id}_{\mathfrak{M}}$ *where* $\circ$ *is the composition of functions and* $\mathsf{Id}_{\mathfrak{M}}$ *is the identity function over set* $\mathfrak{M}$*.*

It is also necessary that the plaintexts and ciphertexts belong to the same set.

Definition 5.6 (Format-Preserving Encryption)**.** *An encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *defined as above is said to be* format-preserving *if* $\mathfrak{M} = \mathfrak{C}$*.*

The encryption scheme is called *in-place* if no additional memory is required for performing encryptions and decryptions.

Definition 5.7 (In-Place Permutations)**.** *A permutation* σ *over set* X *is said to be* in-place *if it can be computed efficiently using exactly the memory registers storing* x *using reversible operations.*

Finally, the encryption scheme is called *non-mixing* if the registers containing the key and auxiliary value are not modified in a way that depends on anything other than themselves. In another sense, the only mixing that is allowed is when modifying the message register during the encryption and decryption process.

Definition 5.8 (Non-Mixing Encryption Scheme)**.** *Let* $(k', \mathrm{aux}', c) = \mathsf{Enc}_k(\mathrm{aux}, m)$ *be the contents of the three memory registers at the end of the in-place encryption algorithm under key* k *(transformed into* k' *by the end of the encryption), where* m *is the message encrypted into ciphertext* c*,* aux *is the auxiliary value and* aux' *is the content of the auxiliary register at the end of the encryption. The encryption scheme is said to be non-mixing if there exists two* in-place *permutations* $e_K : \mathfrak{K} \to \mathfrak{K}$ *and* $e_A : \mathfrak{A} \to \mathfrak{A}$ *such that* $k' = e_K(k)$ *and* $\mathrm{aux}' = e_A(\mathrm{aux})$ *(and similarly for the decryption algorithm with functions* d_K *and* d_A*).*

The function corresponding to the key may represent the key-expansion phase which is often present in symmetric encryption schemes (in which case e_K and d_K are injective functions from $\mathfrak{K}$ to a larger space, but the same results apply), while the one linked to the auxiliary value may be the updating of an initialisation value used in a block cipher mode of operation.

These four previous definitions ensure that the encryption and decryption algorithms can be represented as unitaries when acting on quantum systems without the use of ancillae (which is usual way of transforming a classical function into a quantum operation).

Lemma 5.2 (Sufficient Conditions for Minimal Oracle Representation)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a correct, format-preserving, in-place and non-mixing encryption scheme defined as above (satisfying Definitions 5.5 through 5.8). Then is has a Minimal Oracle Representation according to Definition 5.4. Furthermore for all superpositions* $|\phi\rangle = \sum_{m \in \mathfrak{M}} \alpha_m |m\rangle$ *(where* $|\tilde{\phi}\rangle$ *is the same superposition over encrypted values):*

$$\mathsf{M}_{\mathsf{Enc}} |k\rangle |\mathrm{aux}\rangle |\phi\rangle = |e_K(k)\rangle |e_A(\mathrm{aux})\rangle |\tilde{\phi}\rangle \tag{5.4}$$

Proof. The encryption scheme is correct and format preserving, which implies that, for every key $k \in \mathfrak{K}$ and any value aux $\in \mathfrak{A}$, the functions Enc_k and Dec_k are permutations of $\mathfrak{M}$. In the case of a non-mixing encryption scheme, the functions e_K and e_A are also invertible and so the overall scheme is a permutation over $\mathfrak{K} \times \mathfrak{A} \times \mathfrak{M}$. Any such classical permutations can be represented as minimal oracles: given a permutation σ over a set X it is always possible, although costly, to compute $\sigma(x)$ for all $x \in X$ and define the minimal oracle M_σ by its matrix elements ($\mathsf{M}_\sigma[x][\sigma(x)] = 1$ and 0 everywhere else).

The efficiency of the scheme lies in the fact that all these permutations are in-place, meaning that they can each be computed without using additional memory and using only reversible operations even in the classical case. The classical reversible operations can easily be implemented using unitaries (mainly gates X, CNOT and Toffoli) and so the classical efficiency directly translates to the quantum case.

Finally, since the functions e_K and e_A do not depend on the message being encrypted (or decrypted in the case of d_K and d_A), they remain unentangled from the message register if they were in a basis state prior to the application of the minimal oracle.

■

The Minimal Oracle requirement above forces us to define the symmetric encryption scheme as secure if it is a quantum-secure pseudo-random permutation. In the following, the key-space, auxiliary-space and message-space are fixed to $\mathfrak{K} = \{0,1\}^{n_K}$, $\mathfrak{A} = \{0,1\}^{n_A}$ and $\mathfrak{M} = \{0,1\}^{n_M}$ for some $n_K(\eta)$, $n_A(\eta)$ and $n_M(\eta) > n_K(\eta)$ (mainly for simplicity of presentation, the ideas transpose to other sets). We define the quantum security of such a symmetric encryption scheme by imposing that sampling the key and giving a black-box access to an encryption quantum oracle is indistinguishable for a quantum Adversary from giving it superposition access to a random permutation. This is simply a quantum game-based version of the definition for pseudo-random permutations [60]. For a discussion on this choice of security definitions, see Section 5.4.3.

Definition 5.9 (Real-or-Permutation Security of Symmetric Encryption)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme with Minimal Oracle Representation. Let* $\mathcal{S}_{n_M}$ *be the set of permutations over* $\{0,1\}^{n_M}$*. Consider the following game* Γ *between a Challenger and the Adversary:*

1. *The Challenger chooses uniformly at random a bit* $b \in \{0,1\}$ *and:*
   - *If* $b = 0$*, it samples a key* $k \in \{0,1\}^{n_K}$ *uniformly at random, and sets the oracle* $\mathcal{O}$ *by defining it over the computational basis states* $|\text{aux}\rangle|m\rangle$ *for* $m \in \{0,1\}^{n_M}$ *and* aux $\in \{0,1\}^{n_A}$ *as* $\mathcal{O}|\text{aux}\rangle|m\rangle = U_{\mathsf{Enc}}|k\rangle|\text{aux}\rangle|m\rangle = |k\rangle|e_A(\text{aux})\rangle|\mathsf{Enc}_k(\text{aux}, m)\rangle$ *(the oracle first applies the minimal encryption oracle* $\mathsf{M}_{\mathsf{Enc}}$ *and then the inverse of* d_K *to the register containing the key).*
   - *If* $b = 1$*, it samples a permutation over* $\{0,1\}^{n_M}$ *uniformly at random* $\sigma \in \mathcal{S}_{n_M}$ *and sets the oracle* $\mathcal{O}$ *as* $\mathcal{O}|\text{aux}\rangle|m\rangle = U_{\sigma,e_A}|\text{aux}\rangle|m\rangle = |e_A(\text{aux})\rangle|\sigma(m)\rangle$*.*
2. *For* $i \leq q$ *with* $q = \text{poly}(\eta)$*, the Adversary sends a state* ρ_i *of its choice (composed of* n_M *qubits) to the Challenger. The Challenger responds by sampling an auxiliary value at random* $\text{aux}_i \in \{0,1\}^{n_A}$*, applying the oracle to the state* $|\text{aux}_i\rangle \otimes \rho_i$ *and sending the result back to the Adversary along with the modified auxiliary value (notice that the oracle has no effect on the key if there is one and so it remains unentangled from the Adversary's system).*
3. *The Adversary outputs a bit* $\tilde{b}$ *and stops.*

A symmetric encryption scheme is said to be secure against quantum Adversaries if there exists $\epsilon(\eta)$ negligible in η such that, for any Adversary $\mathcal{A}$ with superposition access and initial auxiliary state ρ_{aux}:

$$Adv_\Gamma(\mathcal{A}) := \left| \frac{1}{2} - \mathbb{P}\Big[b = \tilde{b} \mid \tilde{b} \leftarrow \mathcal{A}(\rho_{aux}, \Gamma) \Big] \right| \leq \epsilon(\eta) \tag{5.5}$$

5.3.1.2 Instantiating the Symmetric Encryption Scheme.

We now show that there exists a concrete encryption scheme satisfying the definitions presented above. In a sense, the perfect (but inefficient) symmetric encryption is given by associating each key $k \in [n_M!]$ to a different permutation from $\mathcal{S}_{n_M}$ in a canonical way (sampling the key is then equivalent to sampling the permutation). The encryption scheme that is used in the protocol may even be considered to be exactly this perfect encryption scheme since the superposition attack does not use the specifics of the underlying encryption scheme, or even supposes a negligible advantage in breaking the encryption scheme (it simply requires it to have a Minimal Oracle Representation).

Note however that a large number of symmetric encryption schemes can be made to fit these conditions. For example, the most widely used block-cipher AES [32] operates by performing during a certain number of rounds N the following process (the message consists of one block of 128 bits, presented a four-by-four square matrix with 8 bits in each cell):

1. It applies a round key (different for each round) by XORing it into the message-block.
2. It applies to each cell a fixed permutation $S : \mathbb{Z}_8 \leftarrow \mathbb{Z}_8$ over 8 bits.
3. Each row i is shifted cyclically by i places to the left (the indices start at 0).
4. Each column is multiplied by a fixed invertible matrix.

This is clearly a permutation of the message block which is also in-place and non-mixing if implemented without optimisations (note that there is no auxiliary value apart from the invertible matrix and the key remains unchanged during the rounds). The security of AES is well studied classically and cryptanalysis has been performed recently in the quantum setting in [18].

The only place where the AES cipher is not in-place is during the key-derivation phase, during which the round keys are generated. The simple way to fix this is to make the register containing the original key large enough to contain the expanded key as well and initialise the additional qubits in the $|0\rangle$ state. The operation producing the larger key from the initial key then corresponds to the functions e_K and d_K from Definition 5.8 (as mentioned in the remark below the definition, these can then be injective only without changing the result).

If the message to be encrypted is longer than a block, it can easily be extended by using the CBC operation mode, which is secure under the assumption that the underlying block-cipher is a quantum-secure pseudo-random permutation (based on the security analysis of [8]). In this mode, the Initialisation Value (or IV) is a uniformly random string of the same size as the blocks upon which the block-cipher operates (it corresponds to the auxiliary value discussed previously). The encryption of the CBC mode operates by applying the function $c_i = \mathsf{Enc}_k(m_i \oplus c_{i-1})$ for all i, where m_i is the message block of index i and c_i is the corresponding ciphertext (with the convention that $c_0 = IV$), and Enc is the encryption algorithm of the underlying block-cipher. Conversely the decryption is given by

$m_i = c_{i-1} \oplus \mathsf{Dec}_k(c_i)$ with the same conventions. This is also clearly in-place and non-mixing (the IV and key are never modified so $e_A = d_A = Id_A$ and $e_K = d_K = Id_K$) as well as secure.

However, as mentioned above, the CBC mode of operation is only secure with superposition access under the assumption that the underlying block-cipher is also secure against Adversaries with superposition access. The cryptanalysis in this setting is much more recent and complex (some schemes previously considered as secure have been broken, such as the 3-round Feistel cipher in [85] and the Even-Mansour cipher in [86], although both were patched in [3] based on the Hidden Shift Problem). On the other hand, the CTR mode of operation on the other hand has been proven quantum-secure in [8] even if the underlying block-cipher is only secure against quantum Adversaries with *classical* access (a more studied setting, although recent attacks by [18] have been found for AES). This mode also satisfies all the requirements stated in the definitions above with only a slight modification. The IV in this mode is also initialised to a uniformly random value of the same size the blocks to be encrypted. The IV is encrypted with the key and XORed to the first block, then the IV in incremented and the same process is repeated for the subsequent blocks. This means that the IV needs to be copied (it can be done quantumly using a CNOT gate but it is then no longer in-place). In order to be in-place and non-mixing, the following procedure may be applied instead for each block (with $\mathcal{IV}$ being the register containing the IV and $\mathcal{B}_i$ the register containing block at position i, with $\leftarrow$ being used for the assignment operator and Inc corresponding to the incrementation operator):

1. $\mathcal{IV} \leftarrow \mathsf{Enc}_k(\mathcal{IV})$
2. $\mathcal{B}_i \leftarrow \mathcal{B}_i \oplus \mathcal{IV}$
3. $\mathcal{IV} \leftarrow \mathsf{Dec}_k(\mathcal{IV})$
4. $\mathcal{IV} \leftarrow \mathsf{Inc}(\mathcal{IV})$

The overall result is that the value contained in the IV has only been updates by incrementing it as many times as there are blocks n_b to be encrypted and therefore the function $e_A = d_A = \mathsf{Inc}^{\circ n_b}$ only acts on the auxiliary value IV.

Note that these methods work for messages whose length is a multiple of the block-size, but can be further extended by using a correctly chosen padding.

5.3.2 Presentation of the Modified Yao Protocol

The protocol will be presented in a hybrid model where both players have access to a trusted third party implementing a 1-out-of-2 String Oblivious Transfer (Ideal Functionality 9). The Garbler plays the role of the Sender of the OT while the Evaluator is the Receiver. The attack presented further below does not rely on an insecurity from the OT (the classical correctness of the Oblivious Transfer is sufficient), which will therefore be supposed to be perfectly implemented and, as all Ideal Functionalities in this model, without superposition access.

Differences with the Original Yao Protocol. There are four main differences between our Modified Yao Protocol 13 and the well-known protocol from [135] recalled in Section 3.5.3. The first two are trivially just as secure in the classical case (as they give no more power to either player): the Garbler sends one copy of its keys to the Evaluator for each entry in the garbled table and instructs it to use a "fresh" copy for each decryption; and the Evaluator returns to the Garbler the copy of the Garbler's keys

that were used in the successful decryption. Notice also that there is only one garbled table for the whole function instead of a series of garbled tables corresponding to gates in the function's decomposition. As explained above, this means that the size of the garbled table is 2^l for inputs of size l (equivalently, this modified protocol can only be used for logarithmically-sized inputs). This is less efficient but no less secure than the original design in the classical case (and quantum case without superposition access), as a player breaking the scheme for this configuration would only have more power if it has access to intermediate keys as well. The last difference is the use of a weaker security assumption for the symmetric encryption function (indistinguishability from a random permutation instead of the quantum equivalents to IND-CPA security developed in [17, 54, 105]). This lower security requirement is imposed in order to model the honest player's actions using the minimal oracle representation. This property influences the security against an adversarial Evaluator, but Theorem 5.2 shows that this assumption is sufficient for security in our scenario. The reasons for these modifications, related to our attack, are developed in Section 5.4.3.

The full protocol for a single bit of output is described formally in Protocol 13.

Protocol 13 Modified Yao Protocol for One Output Bit

Public Information: The function f to be evaluated, the encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ and the size of the padding p.
Inputs: The Garbler and Evaluator have inputs $x \in \{0,1\}^{n_X}$ and $y \in \{0,1\}^{n_Y}$ respectively, with $n_X + n_Y = \mathcal{O}(\log(\eta))$.
Protocol:

1. The Garbler chooses uniformly at random the values $\left\{k_0^{G,i}, k_1^{G,i}\right\}_{i\in[n_X]}, \left\{k_0^{E,j}, k_1^{E,j}\right\}_{j\in[n_Y]}$ from $\mathfrak{K}$ and $k^z \in \{0,1\}$. It uses those values to compute the garbled table $GT_f^{(X,Y,Z)}$, with X being the set of wires for the Garbler's input, Y the set of wires for the evaluators input, and Z the output wire.
2. The Garbler and Evaluator perform n_Y interactions with the trusted third party performing the OT Ideal Functionality. In interaction j:
   - The Garbler's inputs are the keys $(k_0^{E,j}, k_1^{E,j})$, the Evaluator's input is y_j.
   - The Evaluator's output is the key $k_{y_j}^{E,j}$.
3. The Garbler sends the garbled table $GT_f^{(X,Y,Z)}$ and $2^{n_X+n_Y}$ copies of the keys corresponding to its input $\left\{k_{x_i}^{G,i}\right\}_{i\in[n_X]}$. It also sends the auxiliary values $\{\mathrm{aux}_k\}_{k\in[n_X+n_Y]}$ that were used for the encryption of the garbled values.
4. For each entry in the garbled table:
   a) The Evaluator uses the next "fresh" copy of the keys supplied by the Garbler along with the keys that it received from the OT Ideal Functionality to decrypt the entry in the garbled table.
   b) It checks that the last p bits of the decrypted value are all equal to 0. If so it returns the register containing the output value and the ones containing the Garbler's keys to the Garbler.
   c) Otherwise it discards this "used" copy of the keys and repeats the process with the next entry in the garbled table. If this was the last entry it outputs Abort and halts.
5. If the Evaluator did no output Abort, the Garbler applies the One-Time-Pad defined by the key associated with wire z to decrypt the output: if $k^z = 1$, it flips the corresponding output bit, otherwise it does nothing. It then sets the bit in the output register as its output.

The correctness and security in classical networks of this Modified Yao Protocol are captured by

Theorems 5.1 and 5.2, showing that the modifications above have no impact in this setting (against both quantum and classical Adversaries).

Theorem 5.1 (Correctness of the Modified Yao Protocol)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme with a Minimal Oracle Representation (Definition 5.4). Protocol 13 is correct with probability exponentially close to* 1 *in* η *for* $p = \eta$*.*

Proof. We suppose here that both players are honest. Note that the protocol will only fail if one decryption which should have been incorrectly decrypted is instead decrypted as valid. The parameter p must be chosen such that the probability of failure is negligible (in the security parameter in this instance). If at least one of the keys used in decrypting an entry in the garbled table does not correspond to the key used in encrypting it, the encryption and decryption procedure is equivalent to applying a random permutation on $r \parallel 0^p$ for uniformly random r (up to negligible probability in η that the encryption scheme is distinguishable from a random permutation). The probability that the resulting element also has p bits equal to 0 at the end is therefore 2^{-p}.

For $p = poly(\eta)$, we show that the failure probability corresponding to one such event happening across any possible "wrong" decryption is negligible in η. In fact, there are $2^{n_X+n_Y+1}$ ciphertexts (counting both possibilities for k^z) and $2^{n_X+n_Y}$ possible input key combinations, all but one being wrong for each ciphertext. This results in $2^{n_X+n_Y+1}(2^{n_X+n_Y}-1) \approx 2^{2n_X+2n_Y+1}$ random values being potentially generated through incorrect decryption. The probability that none of these random values has the string 0^p as suffix (let Good be the associated event) is given by:

$$\mathbb{P}[\mathsf{Good}] \approx \left(1-2^{-p}\right)^{2^{2n_X+2n_Y+1}} \approx 1 - 2^{-p} \cdot 2^{2n_X+2n_Y+1} \tag{5.6}$$

The first approximation comes from the aforementioned negligible probability that the encryption scheme is not a random permutation while the second stems from $p \gg n_X + n_Y$. This probability should be negligibly close to 1 in η, in which case setting $p = \eta$ is sufficient since $n_X + n_Y = \mathcal{O}(\log(\eta))$.

■

Theorem 5.2 (Classical-Style Security of the Modified Yao Protocol)**.** *Consider a hybrid execution where the Oblivious Transfer is handled by a classical trusted third party. Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme that is* ϵ_{Sym}*-real-or-permutation-secure (Definition 5.9). In classical network* $\mathfrak{C}$*, Protocol 13 is perfectly-secure against adversarial Garbler (the Adversary's advantage is* 0*) and* $(2^{n_X+n_Y}-1)\epsilon_{Sym}$*-secure against adversarial Evaluator according to Definition 5.1.*

Proof. In both cases (adversarial Garbler and Evaluator) we will construct a Simulator that runs the Adversary against the real protocol internally and show that the Environment's advantage in distinguishing the real and ideal executions is negligible. Recall that all exchanged messages are classical.

Security against adversarial Garbler. The Simulator works as follows:

1. During each OT, it performs the same interaction as an honest player would, but with a random value for the input $\tilde{y}_i$ of each OT.

2. The Adversary's machine then necessarily sends the Garbler's keys and the circuit in the computational basis.
3. This automatically fixes the value of the Adversary's input $\hat{x}$ (the Adversary being Honest-but-Curious, it has generated the keys correctly and sent the keys corresponding to its input). The Simulator can therefore measure the register containing the input of the Garbler to recover $\hat{x}$.
4. The Simulator then sends $\hat{x}$ to the Ideal Functionality computing the function f and gets $f(\hat{x}, \hat{y})$ (for the actual value of the honest player's input $\hat{y}$).
5. The Simulator can compute the value $f(\hat{x}, \tilde{y})$ and decrypt the garbled table values to recover the encrypted output $f(\hat{x}, \tilde{y}) \oplus k^z$ using the keys that were giving to it through the OTs (for its "fake" input $\tilde{y}$). It uses both values to recover k^z.
6. The Simulator then computes $f(\hat{x}, \hat{y}) \oplus k^z$ and sends this value to the Adversary.

The only distinguishing advantage of the Environment between the real protocol and this ideal execution stems from the Adversary's potential difference in behaviours during the execution of the OTs. These executions are ideal in the hybrid model and so the advantage of the Environment is 0.

Security against adversarial Evaluator. The messages sent to the adversarial Evaluator consist of n_Y instances of OTs, $2^{n_X+n_Y}$ garbled table entries and the keys corresponding to the input of the honest player. The Simulator performs all these steps similarly to an honest Garbler but sends the keys corresponding to a randomly chosen input $\tilde{x}$. We can show through a series of games that this does not give any information to a computationally-bounded Evaluator (we show that the protocol is input-indistinguishable according to Definition 5.2, which as stated Lemma 5.1 is equivalent since the Adversary has no output):

- **Game 0:** The Simulator performs Protocol 13 with the Adversary, with random input $\tilde{x}$.
- **Game 1:** In the execution of the OTs the Simulator replaces the values of the keys that are not chosen by the Adversary with random values (that were not used to compute any of the encryptions). The advantage in the real-world for the Adversary compared to this situation is 0 since the execution of the OTs is perfectly-secure in the hybrid model.
- **Game 2:** The encryptions that use those (now random) keys can be replaced by random values, with a security cost of ϵ_{Sym} per replaced encryption (as the encryption can be considered to be random permutations without having access to the key). It is a double encryption, so for some values the Adversary may posses either the inner or the outer key. This means that it could invert one of the encryptions, but since it does not have the other this is meaningless.
- **Game 3:** Finally, the key k^z only appears in one encryption as a One-Time-Pad of the output of the computation (the others are now independent from it). It can therefore be replaced by an encryption of a random value, meaning that it is as well a random value (this is perfectly equivalent).

Finally, at the end of Game 3, only the keys received through the OT remain and they are random values chosen independently from one another and from any input. The Environment has no advantage in this scenario, meaning that the overall advantage is at most $(2^{n_X+n_Y} - 1)\epsilon_{Sym}$.

■

The proof above shows that proving the security of some protocols does not require the Simulator to call the Ideal Functionality, in particular if the adversarial party does not have an output in the protocol. This is contrary to the usual simulation-based proofs, where the Simulator must extract the input of the Adversary to send it to the Ideal Functionality (for the sake of composition). However, the exact same proofs of security work in the Stand-Alone Framework of [66] if the Simulator does send the input value of the Adversary to the Ideal Functionality (any Adversary against a classical protocol in the Stand-Alone Framework only sends classical messages as well).

5.4 Superposition Attack on Yao's Protocol

In Section 5.4.1 we first describe how the actions performed during the protocol are transcribed into quantum operations. The superposition attack on the Modified Yao Protocol (Protocol 13) is then presented in two steps: Section 5.4.2 first describes the actions of the Adversary during the execution of the protocol, while Section 5.4.3 presents the Full Attack. This attack is proven to be Extended Honest-but-Curious in Section 5.4.4, therefore the same Adversary in a classical network does not break the classical-style security expressed in Theorem 5.2 (this proves the separation between the quantum and classical settings). The attack is further optimised in Section 5.4.5 using the free-XOR technique, and applied to an Oblivious Transfer protocol (computed by an instance of Yao's Protocol).

Note that this attack does not simply distinguish between the ideal and real executions, but allows the Adversary to extract one bit of information from the honest player's input. It is therefore a concrete attack on the Modified Yao Protocol 13 (as opposed to a weaker statement about not being able to perform an indistinguishable simulation in our model).

5.4.1 Quantum Embedding of the Classical Protocol

The inputs of each party are stored in one register each, as $|x\rangle$ and $|y\rangle$ respectively. For each key k that is created as part of the protocol, a different quantum register is initialised with the state $|k\rangle$ (there are therefore n_Y registers for the Evaluator's keys and $n_X 2^{n_X+n_Y}$ for the Garbler's keys due to the copies being generated). Similarly, for each value E_i of the garbled tables, a quantum register is initialised with the value $|E_i\rangle$ (there are $2^{n_X+n_Y}$ such registers). The auxiliary values are also all stored in separate quantum registers. All of these values are encoded in the computational basis.

The OT trusted party works as described in the Ideal Functionality 9. The inputs and outputs are considered to be pure quantum states in the computational basis (no superposition attack is allowed to go through the OT). Sending messages in the other parts of the protocol is modelled as taking place over perfect quantum channels (no noise is present on the channel and superpositions are allowed to pass undisturbed). A decryption of ciphertext c using a key k and auxiliary value aux is modelled using the Minimal Oracle Representation from Definition 5.4 as $\mathsf{M}_{\mathsf{Dec}} |k\rangle |\text{aux}\rangle |c\rangle = |d_K(k)\rangle |d_A(\text{aux})\rangle |\mathsf{Dec}_k(\text{aux}, c)\rangle$ on the states of the computational basis.

Checking whether the final p bits are equal to 0 corresponds to performing a measurement $\mathcal{M}_C$ on the corresponding register $\mathcal{P}$ in the basis $\{|0^p\rangle\langle 0^p|, 1_{\mathcal{P}} - |0^p\rangle\langle 0^p|\}$. If the measurement fails, the Evaluator applies the inverses of d_K and d_A to the registers containing respectively its keys and the auxiliary values so that they may be reused in the next decryption. Finally, the correction applied at the end which depends on the choice of key for wire Z is modelled as classically controlled Pauli operators X^{k^z}

(this corresponds to the quantum application of a classical One-Time-Pad and the value k^z can be seen as internal classical values of the Garbler for simplicity).

For simplicity, let $k_y^E := k_{y_1}^{E,1} \parallel \ldots \parallel k_{y_{n_Y}}^{E,n_Y}$ for $y \in \{0,1\}^{n_Y}$ (similarly for $x \in \{0,1\}^{n_X}$). Also, let $\widetilde{\mathsf{Enc}}$ be the sequential encryption by all keys corresponding to strings x and y, using the set of auxiliary values $\widetilde{\text{aux}} := \text{aux}_1 \parallel \ldots \parallel \text{aux}_{n_X+n_Y}$. Then $E_{x,y}^{k^z} = \widetilde{\mathsf{Enc}}_{k_x^G, k_y^E}(\widetilde{\text{aux}}, f(x,y) \oplus c \parallel 0^p)$. Finally, $\widetilde{d_K}$ is the function applying d_K to each key, and similarly for $\tilde{d}_A$.

5.4.2 Generating the Correct and Unpolluted Superposition

We start by presenting the action of the adversarial Garbler during the execution of Protocol 13 (its later actions are described in Section 5.4.3). Its aim is to generate a state containing a superposition of its inputs and the corresponding outputs for a fixed value of the Evaluator's input, without it being polluted by additional ancillary registers. This State Generation Procedure on the Modified Yao Protocol 13 (Attack 1) can be summarised as follows:

1. The Adversary's choice of keys, garbled table generation (but for both values of k^z) and actions in the OT are performed honestly.
2. Instead of sending one set of keys as its input, it sends a superposition of keys for two different non-trivial values of the Garbler's input $(\widehat{x_0}, \widehat{x_1})$ (they do not uniquely determine the output).
3. For each value in the garbled table, it instead sends a uniform superposition over all calculated values (with a phase of -1 for states representing garbled values where $k^z = 1$).
4. It then waits for the Evaluator to perform the decryption procedure and, if the Evaluator succeeded in decrypting one of the garbled values and returns the output and register containing the Garbler's keys, the Adversary performs a clean-up procedure which translates each key for bit-input 0 (respectively 1) into a logical encoding of 0 (respectively 1). This procedure depends only on its own choice of keys.

We can now analyse the states of both parties and the success probability of this procedure in Theorem 5.3.

Theorem 5.3 (State Generation Analysis)**.** *The state contained in the Garbler's attack registers at the end of a successful Superposition Generation Procedure (Attack 1) is negligibly close to*

$$\frac{1}{2}\sum_{x,k^z}(-1)^{k^z}\left|x^L\right\rangle\left|f(x,\hat{y})\oplus k^z\right\rangle \tag{5.8}$$

where x^L is a logical encoding of x and $x \in \{\widehat{x_0}, \widehat{x_1}\}$. Its success probability is lower bounded by $1 - e^{-1}$ for all values of n_X and n_Y.

We prove the two parts of Theorem 5.3 separately, first analysing the result of a successful execution and later computing the success probability of the procedure.

Attack 1 Superposition Generation Procedure on the Modified Yao Protocol

Inputs:

- The (adversarial) Garbler has as input the quantum state $\phi_{inp,G} = |\widehat{x_0}\rangle \otimes |\widehat{x_1}\rangle$, with values $\widehat{x_0}, \widehat{x_1} \in \{0,1\}^{n_X}$ received from Environment $\mathcal{Z}$ describing classically the superposition of inputs that it should use.
- The (honest) Evaluator has as input $\hat{y} \in \{0,1\}^{n_Y}$, received from Environment $\mathcal{Z}$.

The Attack:

1. The Garbler chooses uniformly at random the values $\left\{k_0^{G,i}, k_1^{G,i}\right\}_{i\in[n_X]}$ and $\left\{k_0^{E,i}, k_1^{E,i}\right\}_{i\in[n_Y]}$ and computes the *initial* garbled tables $GT_f^{(X,Y,Z),0} = \left\{E_{x,y}^0\right\}_{x,y}$ and $GT_f^{(X,Y,Z),1} = \left\{E_{x,y}^1\right\}_{x,y}$ (where the index 0 corresponds to $k^z = 0$ and similarly for 1, or equivalently that the value encrypted is $f(x,y)$ in the first case and $f(x,y) \oplus 1$ in the second). This computation is the same as in the honest protocol (but done for both values of k^z). Note that there is no need to permute the values as they will be sent in superposition anyway.
2. The Garbler and Evaluator perform n_Y interactions with the trusted third party performing the OT Ideal Functionality. At the end of all interactions, the Evaluator has a quantum register initialised in the state $\bigotimes_{i\in[n_Y]} \left|k_{\hat{y}_i}^{E,i}\right\rangle = \left|k_{\hat{y}}^E\right\rangle$.
3. The Garbler sends the auxiliary values as it would in the original protocol. The corresponding state is $|\widetilde{\text{aux}}\rangle = \bigotimes_{i\in[n_X+n_Y]} |\text{aux}_i\rangle$. For each key k_x^G that it would send to the Evaluator, the Garbler instead sends a uniform superposition $\frac{1}{\sqrt{2}}\left(\left|k_{\widehat{x_0}}^G\right\rangle + \left|k_{\widehat{x_1}}^G\right\rangle\right)$. For each entry in the garbled table that it would send, it instead sends the following superposition over all garbled values:

$$|GT\rangle = \frac{1}{\sqrt{2^{n_X+n_Y+1}}} \sum_{x,y,k^z} (-1)^{k^z} \left|E_{x,y}^{k^z}\right\rangle \tag{5.7}$$

4. For each entry in the garbled table, the Evaluator proceeds as it would in the protocol, decrypting the ciphertexts sequentially, performing a projective measurement on register $\mathcal{P}$ defined by projectors $\{|0^p\rangle\langle 0^p|, 1_{\mathcal{P}} - |0^p\rangle\langle 0^p|\}$ and returning the corresponding output and the register containing the Garbler's keys if successful.
5. If the Evaluator is successful and returns a state after one of its measurements, the Garbler applies the following clean-up procedure:
   a) For each register containing one of its keys, it applied the inverse of d_K.
   b) For each index i such that $\widehat{x_0}^i \neq \widehat{x_1}^i$, if there is an index j such that $k_0^{G,i,j} \neq k_1^{G,i,j}$ and $k_0^{G,i,j} = 1$, it applies an X Pauli operation on the qubit containing this bit of the key.
   c) The first register then contains a superposition of logical encodings of the inputs $\widehat{x_0}^{L'}$ and $\widehat{x_1}^{L'}$. The register containing the output is unchanged.
6. The Garbler then sets these registers (called *attack registers*) as its output, along with a register containing $|\widehat{x_0}\rangle \otimes |\widehat{x_1}\rangle \otimes |\mathbf{L}'\rangle$, with $\mathbf{L}'$ being a list of integers corresponding to the size of a given logical repetition encoding of the inputs (see the proof of Theorem 5.3).

Proof of State Generation Correctness (Theorem 5.3, part 1) The state in the registers of the Evaluator (with input $\hat{y} \in \{0,1\}^{n_Y}$) before it starts the decryption process is (up to appropriate normalisation):

$$|\hat{y}\rangle \otimes \left|k_{\hat{y}}^E\right\rangle \otimes \frac{1}{\sqrt{2}}\left(\left|k_{\widehat{x_0}}^G\right\rangle + \left|k_{\widehat{x_1}}^G\right\rangle\right) \otimes |\widetilde{\text{aux}}\rangle \otimes \sum_{x,y} \left|E_{x,y}^0\right\rangle - \left|E_{x,y}^1\right\rangle \tag{5.9}$$

In fact there are $2^{n_X+n_Y}$ registers containing the superposition of keys and the same number containing the superposition of encryptions, but it suffices to consider the result on one such register (the protocol has been specifically tailored so that repetitions can be handled separately, as seen in the next part of the proof). For $x \neq x'$ or $y \neq y'$ (inclusively), let $g_{x,y}^{x',y',k^z} = \widetilde{\mathsf{Dec}}_{k_x^G,k_y^E}(\widetilde{\text{aux}}, E_{x',y'}^{k^z})$ (this is the decryption of $E_{x',y'}^{k^z}$ using the keys for x and y, leading to a wrong decryption as at least one key does not match and generating the garbage value $g_{x,y}^{x',y',k^z}$). The state after applying the decryption procedure is then (for $x \in \{\widehat{x_0}, \widehat{x_1}\}$):

$$|C\rangle \otimes \left(\sum_{x,k^z} (-1)^{k^z} \left| \widetilde{d_K}\left(k_x^G\right) \right\rangle |f(x,\hat{y}) \oplus k^z\rangle |0\rangle^{\otimes p} + \sum_{\substack{k^z,x,x',y' \\ (x,y)\neq(x',\hat{y})}} (-1)^{k^z} \left| \widetilde{d_K}\left(k_x^G\right) \right\rangle \left| g_{x,\hat{y}}^{x',y',k^z} \right\rangle \right) \tag{5.10}$$

Here the registers containing the Garbler's keys have been rearranged and $|C\rangle = |\hat{y}\rangle \otimes \left|\widetilde{d_K}\left(k_{\hat{y}}^E\right)\right\rangle \otimes \left|\widetilde{d_A}(\widetilde{\text{aux}})\right\rangle$ corresponds to the classical values unentangled from the rest of the state. With overwhelming probability in η (based on the analysis from Theorem 5.1), there are no values (r, k^z, x, x', y') such that the incorrectly decrypted values verify $g_{x,\hat{y}}^{x',y',k^z} = r \parallel 0^p$ and so the states

$$\sum_{x,k^z} (-1)^{k^z} \left| \widetilde{d_K}(k_x^G) \right\rangle |f(x,\hat{y}) \oplus k^z\rangle |0\rangle^{\otimes p} \tag{5.11}$$

and

$$\sum_{\substack{k^z,x,x',y' \\ (x,y)\neq(x',\hat{y})}} (-1)^{k^z} \left| \widetilde{d_K}(k_x^G) \right\rangle \left| g_{x,\hat{y}}^{x',y',k^z} \right\rangle \tag{5.12}$$

are orthogonal. If the measurement $\mathcal{M}_C$ succeeds (i.e. the outcome is $|0^p\rangle\langle 0^p|$), the projected state is (also up to appropriate normalisation):

$$|C\rangle \otimes \sum_{x,k^z} (-1)^{k^z} \left| \widetilde{d_K}(k_x^G) \right\rangle |f(x,\hat{y}) \oplus k^z\rangle |0\rangle^{\otimes p} \tag{5.13}$$

Note that the keys of the Evaluator and the auxiliary values are unentangled from the rest of the state during the whole process thanks to the properties satisfied by the symmetric encryption scheme. The state in the Garbler's registers after receiving the output and its keys is then simply:

$$\sum_{x,k^z} (-1)^{k^z} \left| \widetilde{d_K}(k_x^G) \right\rangle |f(x,\hat{y}) \oplus k^z\rangle \tag{5.14}$$

After applying the first step of clean-up procedure at the end (applying the inverse of d_K for each key), the Garbler is left with the state:

$$\sum_{x,k^z} (-1)^c \left| k_x^G \right\rangle |f(x,\hat{y}) \oplus k^z\rangle \tag{5.15}$$

To demonstrate the effect of the rest of the clean-up procedure, we will apply it to an example with keys $k_0 = 01110$ and $k_1 = 11100$ (for an Adversary's input consisting of a single bit). The corresponding (non-normalised) superposition is then $|k_0\rangle |f(0,\hat{y})\rangle + |k_1\rangle |f(1,\hat{y})\rangle = |01110\rangle |f(0,\hat{y})\rangle + |11100\rangle |f(1,\hat{y})\rangle$ (the terms with $k^z = 1$ behave similarly). If the bits of key are the same, we can factor out the corresponding qubits (in this case, the second, third and fifth qubits are unentangled from the rest). This ends up producing the state $|110\rangle \otimes (|01\rangle |f(0,\hat{y})\rangle + |10\rangle |f(1,\hat{y})\rangle)$. The unentangled qubits may be discarded and then the qubits i for which $k_0^i \neq k_1^i$ and $k_0^i = 1$ are flipped using X (meaning the fourth initial qubit in this case, or the second one after discarding the unentangled qubits). The result is $|00\rangle |f(0,\hat{y})\rangle + |11\rangle |f(1,\hat{y})\rangle$. This procedure does not depend on the choice of $\hat{y}$ (and is the same for $k^z = 1$), only on the keys that were generated by the Adversary.

In the general case, the final clean-up transforms each key associated with a bit-value of 0 into a logical 0 (i.e. $0^{L_i'}$ for a random but known value L_i'), and similarly with the corresponding key associated to the bit-value 1 (changed into $1^{L_i'}$ with the same L_i'). The final result is therefore (where $x^{L'}$ is a logical encoding of x where some bits may be repeated a variable but known number of times):

$$\frac{1}{2}\sum_{x,k^z}(-1)^{k^z}\left|x^{L'}\right\rangle |f(x,\hat{y}) \oplus k^z\rangle \tag{5.16}$$

This is exactly the state that was expected, therefore concluding the proof.

■

Proof of Success Probability of State Generation (Theorem 5.3, part 2) If a given measurement fails, based on the analysis in the previous proof, the state in the Evaluator's registers corresponding to this decryption is negligibly close to:

$$|\hat{y}\rangle \otimes \left|\tilde{d}_K(k_{\hat{y}}^E)\right\rangle \otimes \left|\tilde{d}_A(\widetilde{\text{aux}})\right\rangle \otimes \sum_{\substack{k^z,x,x',y' \\ (x,y)\neq(x',\hat{y})}}(-1)^{k^z}\left|\widetilde{d_K}(k_x^G)\right\rangle \left|g_{x,\hat{y}}^{x',y',k^z}\right\rangle \tag{5.17}$$

By applying the inverse of the d_K and d_A operations on each of the registers containing its keys and the auxiliary values, the Evaluator recovers the state:

$$|\hat{y}\rangle \otimes \left|k_{\hat{y}}^E\right\rangle \otimes |\widetilde{\text{aux}}\rangle \otimes \sum_{\substack{k^z,x,x',y' \\ (x,y)\neq(x',\hat{y})}}(-1)^{k^z}\left|\widetilde{d_K}(k_x^G)\right\rangle \left|g_{x,\hat{y}}^{x',y',k^z}\right\rangle \tag{5.18}$$

Unless it is the last remaining copy of the superposition of Garbler's keys and garbled values (in which case the attack has failed), the Evaluator can simply proceed and repeat the decryption process using its keys and the auxiliary values on the next copy (the failed decryption state is unentangled from the rest and can be ignored in the remaining steps). This essentially means that the Evaluator has $2^{n_X+n_Y}$ independent attempts to obtain measurement result 0^p.

Since the states that are being considered are normalised and in a uniform superposition, the probability of success of each measurement attempt is simply given by the number of states correctly decrypted out of the total number of states.

There are $2^{n_X+n_Y+1}$ encrypted values in the garbled table and 2 key pairs (one key for wires in Y and 2 keys for wires in X). There are therefore $2^{n_X+n_Y+2}$ decrypted values (taking into account decryptions performed with the incorrect keys and counting duplicates). For each key pair, there are exactly two ciphertexts which will decrypt correctly (one for each value of k^z), meaning that 4 decrypted values out of $2^{n_X+n_Y+2}$ have their last p bits equal to 0. The probability of the measurement $\mathcal{M}_C$ succeeding is therefore $\frac{1}{2^{n_X+n_Y}}$. The probability that no measurement succeeds in $2^{n_X+n_Y}$ independent attempts (noted as event Fail) is given by:

$$\mathbb{P}[\mathsf{Fail}] = \left(1 - \frac{1}{2^{n_X+n_Y}}\right)^{2^{n_X+n_Y}} \tag{5.19}$$

The function $p(x) = (1 - \frac{1}{x})^x$ is strictly increasing and upper-bounded by e^{-1}, meaning that the success probability is $\mathbb{P}[\mathsf{Succ}] = 1 - \mathbb{P}[\mathit{Fail}] \geq 1 - e^{-1}$

■

Generalisation to Separable Superpositions. For binary function $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \to \{0,1\}$ and $\hat{y}$, let $U_f^{\hat{y}}$ be the unitary defined through its action of computational basis states by $U_f^{\hat{y}} |x\rangle |k^z\rangle = |x\rangle |f(x,y) \oplus k^z\rangle$. The above procedure allows the Adversary to generate $U_f^{\hat{y}} |\psi\rangle |\phi\rangle$ for any states $|\psi\rangle$ (over n_X qubits) and $|\phi\rangle$ (over one qubit) whose classical descriptions ψ and ϕ are efficient (notice that the state $|\psi\rangle |\phi\rangle$ must be separable). The description of state ψ is used to generate the superposition of keys (if an input appears in the superposition ψ, then the key corresponding to it should appear in the superposition of keys with the same amplitude) while ϕ is used when generating the superposition over garbled table entries, i.e. if $|\phi\rangle = \alpha |0\rangle + \beta |1\rangle$, the corresponding superposition over garbled values is:

$$|GT_{\alpha,\beta}\rangle = \sum_{x,y} \alpha \left|E^0_{x,y}\right\rangle + \beta \left|E^1_{x,y}\right\rangle \tag{5.20}$$

The same results and bounds are applicable (with similar corresponding proofs).

5.4.3 Applying the State Generation Procedure to the Full Attack

We can now analyse the actions of the Adversary after the protocol has terminated. The Full Attack 2 breaking the security of the Modified Yao Protocol 13 can be summarised as follows:

1. The Environment provides the Adversary with the values of the Garbler's input $(\widehat{x_0}, \widehat{x_1})$. The input of the honest Evaluator is $\hat{y}$.
2. The Adversary performs the State Generation Procedure with these inputs.
3. If it has terminated successfully, the Adversary performs an additional clean-up procedure (which only depends on the values of $(\widehat{x_0}, \widehat{x_1})$) to change the logical encoding of $\widehat{x_b}$ into an encoding of b. The resulting state is (omitting this logical encoding, with $b_i := f(\widehat{x_i}, \hat{y})$ and up to a global phase):

$$\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{b_0 \oplus b_1} |1\rangle\right) \otimes |-\rangle \tag{5.21}$$

4. The Adversary recover the XOR of the output values for the two inputs by applying a Hadamard gate to its first register and measuring it in the computational basis.[3]

Attack 2 Full Superposition Attack on the Modified Yao Protocol

---

The Attack:

1. The Environment $\mathcal{Z}$ generates values $(\widehat{x_0}, \widehat{x_1}, \hat{y})$ and sends $|\widehat{x_0}\rangle \otimes |\widehat{x_1}\rangle$ to the Adversary. The values $(\widehat{x_0}, \widehat{x_1})$ are non-trivial in the sense that they do not uniquely determine the value of the output.
2. The Adversary applies the Superposition Generation Procedure described in Attack 1, using a superposition of keys for $\widehat{x_0}$ and $\widehat{x_1}$. If the Evaluator was not successful, the Adversary samples and outputs a bit b (equal to 0 with probability p_{Guess}, which corresponds to the optimal guessing probability and whose value is defined in the proof of Theorem 5.4) and halts.
3. Otherwise, the Adversary applies the following clean-up procedure on the output state of the Superposition Generation Procedure, similar to the one described in Attack 1 (recall that the first register then contains a logical encoding of the inputs $\widehat{x_0}^{L'}$ and $\widehat{x_1}^{L'}$, obtained after the first clean-up procedure described in Attack 1):
   a) If there is an index j such that $\widehat{x_0}^j \neq \widehat{x_1}^j$ and $\widehat{x_0}^j = 1$, it applies a Pauli X operation on the qubits corresponding to the logical encoding of bit j (each bit is encoded with a repetition code of varying length given by the list $\mathbf{L}'$).
   b) The qubits corresponding to a value j such that $\widehat{x_0}^j = \widehat{x_1}^j$ are unentangled from the rest of the state and so can be discarded.
4. The result of the previous step is that the first register now contains a superposition of a logical encoding 0^L and 1^L (for another logical encoding L). The Adversary then applies a logical Hadamard gate H^L on this register.
5. The Adversary measures the first qubit in the computational basis and outputs the result s to $\mathcal{Z}$.
6. The Environment guesses that the execution is real if $s = f(\widehat{x_0}, \hat{y}) \oplus f(\widehat{x_1}, \hat{y})$.

This Full Attack 2 breaks the security of the Modified Yao Protocol 13 (Theorem 5.4) by guessing the XOR of the outputs for two different inputs of the Garbler and the same input for the Evaluator. If the ideal and real executions were indistinguishable according to Definition 5.1, such a feat would be impossible for the Adversary since the Simulator can at most access one value of the output through the Ideal Functionality.

Theorem 5.4 (Vulnerability to Superposition Attacks of the Modified Yao Protocol)**.** *For any non-trivial two-party function $f : \{0,1\}^{n_X} \times \{0,1\}^{n_Y} \to \{0,1\}$, let $(\widehat{x_0}, \widehat{x_1})$ be a pair of non-trivial values in $\{0,1\}^{n_X}$. For all inputs $\hat{y}$ of honest Evaluator in Protocol 13, let $\mathsf{P}_f^E(\hat{y}) = f(\widehat{x_0}, \hat{y}) \oplus f(\widehat{x_1}, \hat{y})$. Then there exists a real-world Adversary $\mathcal{A}$ in quantum network $\mathfrak{Q}$ against Protocol 13 implementing f such that for any Simulator $\mathcal{S}$, the advantage of the Adversary over the Simulator in guessing the value of $\mathsf{P}_f^E(\hat{y})$ is lower-bounded by $\frac{1}{2}(1 - e^{-1})$.*

Proof. Let $(\widehat{x_0}, \widehat{x_1})$ be a pair of values in $\{0,1\}^{n_X}$ such that there exists $(\widehat{y_0}, \widehat{y_1})$ with $f(\widehat{x_0}, \widehat{y_0}) = f(\widehat{x_1}, \widehat{y_0})$ and $f(\widehat{x_0}, \widehat{y_1}) \neq f(\widehat{x_1}, \widehat{y_1})$ (at least one such pair of inputs exists, otherwise the function is trivial). The Environment $\mathcal{Z}$ initialises the input of the Adversary with values a pair of such values $\widehat{x_0}$ and $\widehat{x_1}$. Let $\hat{y} \in \{0,1\}^{n_Y}$ be the value of the honest player's input chosen (uniformly at random) by the Environment $\mathcal{Z}$. The goal of the attack is to obtain the value of $\mathsf{P}_f^E(\hat{y}) = f(\widehat{x_0}, \hat{y}) \oplus f(\widehat{x_1}, \hat{y})$.

[3] This corresponds to the final steps of the DJ algorithm after the application of the oracle, see Section 2.2.3.1.

The Adversary will try to generate the superposition state during the protocol using Attack 1, succeeding with probability p_{Gen}. If the state has been generated correctly, Adversary will apply the final steps of Deutsch's algorithm and recover the value of the XOR with probability equal to 1 (see below). If the state generation fails, Adversary resorts to guessing the value of the value of $\mathsf{P}_f^E(\hat{y})$, winning with a probability p_{Guess}. On the other hand, the Simulator is only able toss a coin to guess the value of $\mathsf{P}_f^E(\hat{y})$ (the only information that it possesses is either $f(\widehat{x_0}, \hat{y})$ or $f(\widehat{x_1}, \hat{y})$, given by the Ideal Functionality), winning with probability p_{Guess}.

The overall advantage of the Adversary is therefore $p_{Gen} \cdot (1 - p_{Guess})$ (if the State Generation Procedure does not succeed, the probabilities of winning of the Adversary and the Simulator are the same). It has been shown via Theorem 5.3 that the probability of generating the state is lower-bounded by $1 - e^{-1}$, the rest of the proof will focus on describing the last steps of the Attack 2 and calculating the other values defined above.

We first analyse the behaviour of the state during the Adversary's calculation in Attack 2 if there was no Abort. The state in the register of the Adversary registers at the end of a successful Superposition Generation Procedure via Attack 1 is (the logical encoding L' being known to the Adversary):

$$\frac{1}{2}\Big(\Big|\widehat{x_0}^{L'}\Big\rangle |f(\widehat{x_0}, \hat{y})\rangle - \Big|\widehat{x_0}^{L'}\Big\rangle |f(\widehat{x_0}, \hat{y}) \oplus 1\rangle + \Big|\widehat{x_1}^{L'}\Big\rangle |f(\widehat{x_1}, \hat{y})\rangle - \Big|\widehat{x_1}^{L'}\Big\rangle |f(\widehat{x_1}, \hat{y}) \oplus 1\rangle\Big) \tag{5.22}$$

The Adversary applies the clean-up procedure on the registers containing $\widehat{x_i}^{L'}$ and obtains (for a different value L for the logical encoding):

$$\frac{1}{2}\big(|0\rangle^{\otimes L} |f(\widehat{x_0}, \hat{y})\rangle - |0\rangle^{\otimes L} |f(\widehat{x_0}, \hat{y}) \oplus 1\rangle + |1\rangle^{\otimes L} |f(\widehat{x_1}, \hat{y})\rangle - |1\rangle^{\otimes L} |f(\widehat{x_1}, \hat{y}) \oplus 1\rangle\big) \tag{5.23}$$

This is exactly the state of Deutsch's algorithm after applying the (standard) oracle unitary implementing $U_{f_{\hat{y}}^{\widehat{x_0},\widehat{x_1}}}$, where $f_{\hat{y}}^{\widehat{x_0},\widehat{x_1}}(b) = f(\widehat{x_b}, \hat{y})$ (by standard we mean of the form $U_f |x\rangle |b\rangle = |x\rangle |b \oplus f(x)\rangle$, in comparison to the Minimal Oracle Representation). The rest of the attack and analysis follows the same pattern as Deutsch's algorithm. For simplicity's sake, let $b_i := f(\widehat{x_i}, \hat{y})$, then the state is:

$$\frac{1}{\sqrt{2}}(-1)^{b_0}\big(|0\rangle^{\otimes L} + (-1)^{b_0 \oplus b_1} |1\rangle^{\otimes L}\big) \otimes |-\rangle \tag{5.24}$$

The Adversary then applies the logical Hadamard gate, the resulting state is (up to a global phase):

$$|b_0 \oplus b_1\rangle^{\otimes L} \otimes |-\rangle \tag{5.25}$$

The Adversary can measure the first qubit in the computational basis and distinguish perfectly both situations, therefore obtaining $f(\widehat{x_0}, \hat{y}) \oplus f(\widehat{x_1}, \hat{y}) = b_0 \oplus b_1$.

On the other hand, in the ideal scenario, to compute the probability p_{Guess} of guessing the correct answer, we consider the mixed (i.e. probabilistic) strategies in a two-player game between the Environment $\mathcal{Z}$ and the Simulator where both players choose a bit simultaneously, the Simulator wins if they are the same and the Environment wins if they are different. This represents the most adversarial Environment

for the Simulator. The goal is to find a Pareto equilibrium in this game. [4] The Environment chooses bit-value 0 with probability p, while the Simulator chooses the bit-value 0 with probability q. The probability of winning for the Simulator is then $p_{Guess} = pq+(1-p)(1-q) = 1-q-p(1-2q)$. We see that if $q \neq 1/2$ there is a pure (i.e. deterministic) strategy for the Environment such that $p_{Guess} < 1/2$ (if the Simulator chooses its bit in a way that is biased towards one bit-value, the Environment always chooses the other), while if $q = 1/2$ then $p_{Guess} = 1/2$. The same analysis can be applies to the Environment and therefore $p = 1/2$ as well.

In the end, we have $p_{Guess} = 1/2$ and therefore the advantage of the Adversary is $\mathsf{Adv} = p_{Gen}(1 - p_{Guess}) \geq \frac{1}{2}(1 - e^{-1})$, which concludes the proof.

■

As a remark, the inverse of the circuit preparing the GHZ state can be applied as the final step of the attack instead of the logical Hadamard H_L, yielding the same result (the state is then $|b_0 \oplus b_1\rangle \otimes |0\rangle^{\otimes L-1}$ and measuring the first qubit in the computational basis still gives the correct value). The presentation above was chosen to closely reflect the description of Deutsch's algorithm.

Justifying the Differences in the Protocol Variant. We can now more easily explain the choices straying from the Original Yao Protocol mentioned in Section 5.3.2. The first remark is that the fact that the Garbler sends multiple copies of its keys is what allows the success probability to be constant and independent from the size of the inputs (see Theorem 5.3). Otherwise it would decrease exponentially with the number of entries in the garbles table, which might not be too bad if it is a small constant (the minimum being 4 for the naive implementation). On the other hand, returning the Garbler's keys to the Adversary is an essential part of the attack, as otherwise it would not be able to correct them (the final operations described in the full attack are all performed on these registers). If they stay in the hands of the Evaluator, it is unclear how the Adversary would perform the attack (as the state is then similar to the entangled one described in the introduction as something that we seek to avoid). Similarly, the fact that we do not use an IND-CPA secure symmetric encryption scheme is linked to the fact that it adds an additional register containing the randomness used to encrypt (for quantum notions of IND-CPA developed in [17] and [105]), and which is then entangled with the rest of the state (this register can not be given back to the Adversary as it would break the security, even in the classical case, by revealing the index of the correctly decrypted garbled entry). On the other hand, in [54] they show that the notion of quantum IND-CPA they define is impossible for *quasi-length-preserving* encryption scheme (which is equivalent to *format-preserving* from Definition 5.6). Finally, if we were to follow the same principle as in the original protocol and decompose the binary function into separate gates, then the intermediate keys would similarly add another register which is entangled with the rest of the state. This is why we require that the garbled table represents the whole function.

5.4.4 The Full Attack is not Malicious

Note that the Original Yao Protocol is secure against Honest-but-Curious Adversaries. The equivalent in terms of superposition attacks is to send exactly the same messages but computed over an arbitrary superposition of the randomness used by the Adversary (be it the inputs of other random values). That

[4] These equilibria are modelled as strategies for both players such that none can gain more by deviating unilaterally.

is to say that, if the honest party would have measured the state sent by the Adversary, it would recover perfectly honest classical messages. On the other hand, the Adversary described in Attack 2 is not strictly Honest-but-Curious.

However, the following lemma captures the fact that the previously described Adversary does not break the Honest-but-Curious security of the Modified Yao Protocol if it does not have superposition access (a fully-malicious one can trivially break it), thereby demonstrating the separation between Adversaries with and without superposition access.

Lemma 5.3 (Adversarial Behaviour Analysis)**.** *In a classical network* $\mathfrak{C}$*, the Adversary described in Attack 2 is an* Extended Honest-but-Curious Adversary *(Definition 5.3).*

Proof. Attack 2 is not strictly Honest-but-Curious since a player that measures honestly and tries to decrypt after can also fail with probability e^{-1} if it never gets the correct ciphertext in the table after measuring. When restricted to classical networks, this Adversary works as follows:

1. It generates all values for the garbled table (for both values of k^z).
2. For each garbled table entry that it is supposed to send, it instead chooses uniformly at random one of the generated values (with replacement) and a key for either $\widehat{x_0}$ or $\widehat{x_1}$ and sends them (it does not store in memory which values have been sent).
3. It then waits to see if the honest player has been able to decrypt one of the values or not.
4. If it has, then it receives (as classical messages) the key that was used to decrypt (either for $\widehat{x_0}$ or $\widehat{x_1}$) and the decrypted value.

This Adversary is precisely an Extended Honest-but-Curious Adversary according to Definition 5.3 as the Simulator presented in the security proof of Theorem 5.2 works as well for this Adversary, with the difference that with a probability of e^{-1} it cannot recover the value of k^z if it is unable to decrypt (but then this is also the case when interacting with an honest party) and so must abort. Since the Adversary does not store which values have been sent it does not know whether this value has been decrypted from the keys from the honest player or the Simulator (using a random input). On the other hand this action by the Simulator is necessary to simulate the probability that none of the keys decrypt correctly the garbled values (this happens with the same probability in the simulated and real executions).

■

The core reason why the Honest-but-Curious Simulator works is that the Adversary's internal register is never entangled with the states that are sent to the honest party: much more efficient attacks exist in that case, for example the Adversary can recover the full input of the Evaluator if it keeps a register containing the index of the garbled table value, which collapses along with the output register when it is measured by the honest player while checking the padding, therefore revealing the input of the Evaluator. However this Adversary is not simulatable when placed in a classical network (and therefore this attack does not show a separation between the two scenarii as it would be similar to subjecting the protocol to a Malicious classical Adversary, that can trivially recover the honest player's input).

5.4.5 Attack Optimisation and Application to Oblivious Transfer

The attack described in Section 5.4 will now be applied to a simple function, namely the 1-out-of-2 bit-OT, in order to demonstrate a potential improvement. In this case, the Garbler has a bit b as input,

the Evaluator has two bits (x_0, x_1) and the output for the Garbler is x_b. This can be represented by the function $\mathcal{OT}(b, x_0, x_1) = bx_1 \oplus (1 \oplus b)x_0$. This can be factored as $\mathcal{OT}(b, x_0, x_1) = b(x_0 \oplus x_1) \oplus x_0$. By changing variables and defining $X := x_0 \oplus x_1$, it can be rewritten further into $\mathcal{OT}(b, x_0, X) = bX \oplus x_0$.

Based on this simplified formula, instead of computing the garbled table for the full function, the Garbler will only garble the AND gate between b and X. In order to compute the XOR gate at the end, the Free-XOR technique will be used. Recall first that the key-space is fixed to $\mathfrak{K} = \{0,1\}^{n_\kappa}$. Instead of choosing both keys for each wire uniformly at random, this technique works by choosing uniformly at random a value $K \in \{0,1\}^{n_\kappa}$ and setting $k_1^w := k_0^w \oplus K$ for all wires w which are linked to the XOR gate (either as input or output wires). The value k_0^w is sampled uniformly at random for the input wires. For the output wire, if a and b are the labels of the input wires, the value is set to $k_0^w = k_0^a \oplus k_0^b$. In this way, instead of going through the process of encrypting and then decrypting a garbled table, given a key for each input of a XOR gate, the Evaluator can directly compute the output key in one string-XOR operation (as an example, if the keys recovered as inputs for the input wires are k_0^a and k_1^b, then the output key is computed as $k_0^a \oplus k_1^b = k_0^a \oplus k_0^b \oplus K = k_0^w \oplus K = k_1^w$, which is the correct output key value for inputs $a = 0$ and $b = 1$). The security of Yao's protocol using the Free-XOR technique derives from the fact that only one value for the keys is known to the evaluator at any time, so the value K is completely hidden (if the encryption scheme is secure). This has been first formalised in [83].

After having decrypted the garbled table for the AND gate, the Evaluator simply performs the XOR gate using the Free-XOR technique. Without loss of generality the XOR of the keys is performed into the register containing the key corresponding to the output of the AND gate. In the quantum case, this is done using a CNOT gate, where the control qubit is the register containing the keys for x_0 and the controlled qubit is the register containing the output of the decryption of the garbled AND gate (the key for x_0 is not in superposition as it belongs to the Evaluator and so the register containing it remains unentangled from the rest on the state).

The initial input to the garbled table is 3 bits long in the decomposed protocol, while the input to the AND gate is only 2 bits long, lowering the number of pre-computations to generate the garbled table and improving slightly the attack's success probability (it is a decreasing function of the number of possible inputs).

The probability of successfully generating the attack superposition $\frac{1}{2}\big(|0\rangle^{\otimes L}|x_0\rangle - |0\rangle^{\otimes L}|x_0 \oplus 1\rangle + |1\rangle^{\otimes L}|x_1\rangle - |1\rangle^{\otimes L}|x_1 \oplus 1\rangle\big)$ by using this new technique is $1 - \left(\frac{3}{4}\right)^4 = \frac{175}{256}$ (by not using the approximation at the end of the proof of part 2 of Theorem 5.3 for success probability). As described in Theorem 5.4, such a superposition can been used to extract the XOR of the two values, an attack which is impossible in the classical setting or even in the quantum setting without superposition access. The advantage of the Adversary in finding the XOR (over a Simulator which guesses the value) by using this attack is $\frac{175}{512}$. This is far from negligible and therefore the security property of the OT is broken.

Of course this is a toy example as it uses two *string*-OTs to generate one *bit*-OT. But the bit-OT that has been generated has a reversed Sender and Receiver compared to the string-OTs that were used. In the classical case, it can be noted that similar constructions have been proposed previously to create an OT which was simulatable for one party based on an OT that is simulatable for the other (and this construction is close to round-optimal).

5.5 Security Model Satisfiability

Having dissected an attack on a protocol, we now give feasibility results in our model. As a cryptographic "Hello World", we first prove in Section 5.5.1 that the classical One-Time-Pad remains secure even against superposition attacks. Section 5.5.2 then analyses post-mortem the superposition attack on Yao's Protocol to build a Superposition-Resistant Yao Protocol.

5.5.1 Superposition-Resistance of the Classical One-Time Pad

The Classical OTP (Protocol 14) uses a Key Distribution (Ideal Functionality 7, see [114]) for two parties to emulates a Confidential Channel (Ideal Functionality 2), which assures that only the length of the message is leaked to the Eavesdropper but does not guarantee that it was not tampered with (see also [41] and [28]). The security of the Classical OTP Protocol against Adversaries in classical networks $\mathfrak{C}$ is proven in [41].

Protocol 14 Classical OTP Protocol

Inputs: The Sender has a message $m \in \{0,1\}^n$. The Receiver has as input the size of the message. The Eavesdropper has an auxiliary input ρ_{aux}.

Protocol:

1. The Sender and Receiver call the Key Distribution Ideal Functionality on input n and receive a key k of size n.
2. The Sender computes $y = m \oplus k$ (where $\oplus$ corresponds to a bit-wise XOR) and sends it to the Eavesdropper.
3. The Eavesdropper sends a message $\hat{y}$ to the Receiver.
4. The Receiver compute $\hat{m} = \hat{y} \oplus k$

We will now prove the security of the protocol against malicious Receiver in quantum network $\mathfrak{Q}$ (with superposition access), as captured by the following Lemma 5.4.

Lemma 5.4 (Security of One-Time-Pad against Adversaries with Superposition Access)**.** *Protocol 14 is superposition-resistant against a malicious Eavesdropper with advantage $\epsilon = 0$ (i.e. it satisfies Definition 5.1 in quantum network $\mathfrak{Q}$).*

Proof. We start by defining the quantum equivalent of all operations in Protocol 14. The initial message is represented as a quantum register containing $|m\rangle$. The call to the Key Distribution Ideal Functionality yields a quantum register for both parties containing a state $|k\rangle$ in the computational basis. The bit-wise XOR is applied using CNOT gates where the key corresponds to the control. The definition of the CNOT gate implies that if the control is in a computational basis state, it remains unentangled from the rest of the state after application of the gate. The state is then sent to the Eavesdropper. It can perform any CPTP map on the state $|y\rangle \otimes \rho_{aux} \otimes |0^{\otimes n}\rangle$ and send the last register to the Receiver. The Receiver applies the XOR using CNOT gates with its key as control.

The Eavesdropper has no output in this protocol. As stated in Lemma 5.1, it would be sufficient to show that two executions with different inputs are indistinguishable. However we will now describe the Simulator for clarity. It receives the size of the message n from the Confidential Channel Ideal Functionality. It chooses uniformly at random a value $\tilde{y} \in \{0,1\}^n$ and sends $|\tilde{y}\rangle$ to the Eavesdropper. It

receives in return a state ρ on n qubits and sends it to the Confidential Channel Ideal Functionality (which then measures the state in the computational basis).

Before the message sent by the Adversary, the protocol is equivalent to its classical execution, so the Environment has no additional advantage compared to the classical execution (which is perfectly secure). The only advantage possibly obtained by the Adversary compared to a fully classical one comes from the state that it sent to the Receiver (repsectively Simulator) and the application by the Receiver of an operation dependent on its secret key (respectively a measurement in the computational basis by the Ideal Functionality). It is a well known fact (No-Communication Theorem of quantum information) that the Environment obtaining any bit of information with probability higher than 0 via this method (using only local operation on the Receiver's side or by the Ideal Functionality) would violate the no-signalling principle [59, 47], therefore the distinguishing advantage of the Environment between the real and ideal executions is 0, thereby concluding the proof.

■

5.5.2 Superposition-Resistant Yao Protocol

We can now analyse the crucial points where the security breaks down and propose counter-measures. We notice that all actions of the Adversary only act on the registers that contain its own keys (recall that the Evaluator sends back the Garbler's keys after a successful decryption) and have no effect on the output register, which stays in the $|-\rangle$ state the whole time. It is thus unentangled from the rest of the state and the attack on the protocol can therefore also be performed if the Garbler has no output. As the security in this case still holds for Adversaries in classical network $\mathfrak{C}$ via input-indistinguishability, it means that this security property does not carry over from the classical to the quantum network case either.

Therefore, as counter-intuitive as it may seem, the precise point that makes the attack possible is a seemingly innocuous message consisting of information that the Adversary should (classically) already have, along with a partial measurement on the part of the honest player (which is even stranger considering that it is usually thought that the easiest way to prevent superposition attack is to measure the state).

Not sending back this register to the Adversary (as in the Original Yao Protocol) makes the protocol structurally similar to the One-Time-Pad Protocol 14: one party sends everything to the other, who then simply applies local operations. The proof for the One-Time-Pad works by showing that there is a violation of the no-signalling condition if the Environment is able to guess whether it is in the real or ideal situation. This technique can be reused if the Evaluator does not give away the result of the measurement on its state (by hiding the success or failure of the garbled table decryption[5]).

We give here a sketch of the formal Superposition-Secure Yao Protocol 15, along with a proof of its security against an adversarial Garbler with superposition access. It uses Yao's original efficient construction for the garbled table, where the function is decomposed into elementary gates (of constant fan-in) and can therefore be applied to any binary function with inputs that are of polynomial size in the security parameter (see [89] for the construction).

[5]This contradicts the footnote in Section 5.2 before Ideal Functionality 19 since the proof works if there is no future communication between the two players.

Protocol 15 Superposition-Secure Yao Protocol (Sketch)

Inputs: The Garbler and Evaluator have inputs $x \in \{0,1\}^{n_X}$ and $y \in \{0,1\}^{n_Y}$ respectively, with $n_X + n_Y = poly(\eta)$.
Public Information: The function f to be evaluated, the encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ and the size of the padding p.
Protocol:

1. The Garbler creates the keys and garbled table as in the original Yao's Protocol (with no k^z and using the function's decomposition into AND and OR gates).
2. The Garbler and the Evaluator participate in the OT ideal executions, at the end of which the Evaluator receives its evaluation keys for its input of choice.
3. The Garbler sends the evaluation keys for its inputs and stops.
4. The Evaluator decrypts each entry in the garbled table sequentially. It stops if the padding is 0^p, the first bit is then set as its output.
5. Otherwise (if none of the values were decrypted correctly), it sets as its output Abort. This is not communicated to the Garbler.

Theorem 5.5 shows that Protocol 15 is secure both in quantum and classical networks.[6]

Theorem 5.5 (Security of Superposition-Resistant Yao Protocol in Quantum Network $\mathfrak{Q}$)**.** *In quantum network $\mathfrak{Q}$, the Superposition-Resistant Yao Protocol 15 is perfectly-secure against an adversarial Garbler according to Definition 5.1 in an OT-hybrid execution.*

Proof (Sketch) The Garbler cannot break the security of the OT ideal execution, which furthermore is classical. The rest of the protocol can be summarised by the Garbler sending one quantum state and then the Evaluator performing a local operation on it and stopping. This is exactly the same scenario as in the One-Time Pad protocol and the same analysis applies in this case.

The Simulator uses a random input during the ideal executions of the OT (which it controls and is classical). It then receives a state ρ corresponding to the Garbler's keys and garbled table. The only advantage possibly obtained by the Adversary compared to one in a classical network comes from this state and the application by the Evaluator of the decryption procedure using its secret keys and those of the Garbler (compared to no operations by the Simulator). The No-Communication Theorem of quantum information implies that the Environment obtaining any bit of information with probability higher than 0 via this method (using only a local operation on the Evaluator's side) would violate the no-signalling principle [59, 47], therefore the distinguishing advantage is 0.

■

The proof above does not translate into a proof for an actual instance of the protocol since security in our model does not hold under sequential composability, but it gives a hint as to which steps are crucial for securing it. Another path for obtaining security could be to replace the encryption scheme with one for which there is no efficient Minimal Oracle Representation. We leave this case as an open question.

[6] As noted in Section 5.2, superposition-resistance implies classical-style security.

5.6 Conclusion and Discussion

Our security model and the attack analysis performed in our work lie completely outside of the existing models of security against superposition attacks. They either consider the computational security of basic primitives or, for more complex protocols with multiple interactions between distrustful parties, the protocols are all considered to be statistically-secure (and are therefore essentially extensions of [100]). This leads to many simplifications which have no equivalent in the computational setting. We develop a novel security framework, based on the simple premise that to be secure from superposition attacks means emulating a purely classical functionality. We show that, given slight modifications that preserves classical security, it is possible to show superposition attacks on computationally-secure protocols. The intuition gained from the attack allows us to build a computationally superposition-resistant protocol for Two-Party Secure Function Evaluation, a task never achieved before.

Our results demonstrate once again the counter-intuitive nature of quantum effects, regarding not only the vulnerability of real-world protocols to superposition attacks (most would require heavy modifications for known attacks to work), but also attack vectors and the optimal ways to counter them (as partial measurements can even lead to attacks).

Future Work

An interesting research direction would be to analyse what functionalities (if any) can be implemented using the "insecure" ideal functionalities with allowed superposition access described in [122]. Since these functionalities necessarily leak information, they can no longer be universal: if they were, then it would be possible to construct non-leaky functionalities with protocols only making calls to these leaky functionalities. However, some limited functionalities may also be useful, as exemplified by the biased coin toss.

The security model presented in this chapter does not support any kind of composability, as can be shown with rather simple counter-examples. While it would be ideal to have a simulation-based fully-composable framework for security against superposition attacks, we leave this question open for now.

While we prove that Yao's protocol is secure in our model if the Evaluator does not reveal the outcome of the protocol, it would also be interesting to analyse the consequence of removing the minimal oracle assumption from the symmetric encryption scheme and instead use a traditional IND-CPA symmetric encryption with the original Yao garbled table construction (therefore adding an additional entangled quantum register). The Yao protocol has recently been studied in [24] and found secure against Adversaries that do not have superposition access to the honest party, under the assumption that the encryption scheme is pq-IND-CPA (the quantum Adversary does not make queries to the encryption oracle in superposition but has access to a Quantum Random Oracle).

Finally, our result shows that partial measurements by honest players are not sufficient to prevent superposition attacks. It would be interesting to find the minimum requirements for the security of protocols with superposition access and measurements by honest parties so that they are as secure as classical protocols. This field of study has been somewhat initiated by the work of [129] with the collapsing property (measuring one message makes the other message collapse to a classical value if it

passes some form of verification), but the question of whether there is a minimal amount of information that should be measured to be superposition-secure remains open.

Chapter 6

Quantum Round-Optimal Protocol for Delegated Multi-Party Quantum Computations versus Dishonest Majority

6.1 Motivation and Overview of Results

6.1.1 Delegation, Distribution and Composition

Most quantum computers are being designed to be accessible through cloud services. Out of convenience and efficiency for end users, it is expected to be the main way for relatively powerless clients to access highly powerful quantum servers: the existence of several schemes for delegating quantum computations from a single client to a single server supports the idea [118, 4, 36]. Such schemes can be made even more appealing and adapted to reality by requiring secure delegation. This means that the client could require confidentiality of the algorithm and data sent to the server together with integrity of the computation itself. Various protocols have been developed to provide solutions for diverse settings and security levels [21, 53, 72, 78, 58, 52]. For instance, the client could be completely classical or have single gate capabilities and a quantum channel; the security could be information theoretical or only computational, etc.

Regarding Multi-Party Quantum Computation, several lines of research have been followed during the past two decades. The very first protocol was developed in [30]. Along with the introduction of the concept itself, they provided a concrete protocol for performing such computations in the quantum circuit model. It guarantees the security of the computation as long as the fraction of malicious parties does not exceed 1/6. This work has been later extended in [14], lowering the minimum number of honest players required for security to a strict majority.

Another direction was studied in parallel regarding the possible composability of such protocols, as earlier results did not satisfy this property. Bit commitment was shown to be complete in the Quantum

Universal Composability framework of [126], meaning that it is sufficient for constructing quantum or Classical SMPC if parties have access to quantum channels and operations. This result was later extended in [49, 42], which give a full analysis of feasibility and completeness of cryptographic primitives in a composable setting.

More recently, building on these three branches of study, new concrete protocols have been proposed to decrease the restrictions on adversaries and provide composable security. First, [76] described a protocol that is composable, can tolerate a dishonest majority and allows the clients to delegate the quantum computation to a powerful server. Its security is an information-theoretic upgrade of a Classical SMPC primitive used for constructing the protocol. It is however limited by the absence of verifiability of outputs and the impossibility to tolerate client-server collusion. Reference [69] provides a generic recipe to turn any composable two-party delegated blind and verifiable quantum computation protocol into a multi-party delegated version. It can cope with a dishonest server only, requires a complete quantum and classical communication graph and is limited in terms of implementable computations as each client chooses only local computations and cannot coordinate with others. In the circuit model with teleportation for magic states, a composably-secure protocol has been introduced recently in [40]. It is an extension of [45] that is able to cope with a dishonest majority, but which relies on a complete graph for quantum communication architecture that imposes a large number of quantum communication rounds together with powerful quantum participants. A stand-alone secure protocol is proposed in [90] as an extension on previous results of [14] based on error-correcting codes. Its aim is to lower the amount of quantum memory of participants but suffers from the same drawback as [40]: powerful quantum parties and complete quantum and classical communication graph.

With the exception of [76], which provides only blindness in absence of clients-server collusions, these proposed solutions have mostly eschewed the interesting situation of delegated multi-party quantum computations to focus rather on a symmetrical setting where each party has an equally highly powerful quantum computer. This left open the question of the existence of efficient delegated MPQC protocols in a fully adversarial scenario.

6.1.2 Our Contribution

Our protocol closes this gap by demonstrating an efficient delegated MPQC construction for clients with limited quantum capabilities, secure against a malicious collusion between any number of clients and the server, thus recovering all the previous advantages of protocols in the symmetric setting and improving vastly the quantum communication overhead. As in the symmetrical case [40], it lifts a secure Classical Secure Multi-Party Computation to the quantum realm in an information-theoretical way, meaning that the protocol offers the same security level as the Classical SMPC used in its construction. The security is proven in the fully-composable Abstract Cryptography Framework [99] (presented in Section 3.3.2), hence the security guarantees of the protocol hold not only for standalone uses but also in situations where it is employed many times, sequentially or in parallel, with or without being included as a part of a broader secure task.

Based on the single-client construction of [78], the clients in our protocol only need to be able to manipulate single qubits and perform *two* rounds of quantum communication with the server (reduced to a single round if the outputs are classical), which is in fact optimal. It inherits also the delegated

nature of this protocol, meaning that only a single party needs to perform elaborate quantum operations while the others need only to be able to do single qubit operations and state generation.

High-Level Protocol Construction. To achieve these results, several new techniques need to be introduced. We apply a deconstruction-reconstruction approach to the Delegated MPQC problem, in synergy with the top-down methodology of AC, to identify smaller key functionalities that needed to be replaced to successfully turn the VBQC Protocol of [78] into a multi-party one that could be driven by a Classical SMPC. The VBQC (and UBQC) Protocol can be decomposed into the following steps: (i) VBQC client-encrypted state preparation, (ii) graph-state entanglement, (iii) rotated measurement of non-outputs (iv) output recovery and decryption. The proposed solution is then obtained by replacing some of these steps of the deconstructed VBQC Protocol in order to cope with multiple clients. More precisely, step (i) is modified so that the VBQC client-encrypted state used in the VBQC protocol is now obtained as the result of a collaborative computation delegated to the server. This has led us to define and construct a new quantum resource, called Double Blind Quantum Computation (DBQC). It allows a fully classical trusted party – an *Orchestrator* – to drive a quantum computation (the description of which is known only to this Orchestrator) between several clients that provide inputs and a server. This is used to prepare the VBQC client-encrypted state collaboratively. For steps (ii-iv) to be completed, all single-client classical driving of the server's action need now to be replaced by a collaborative driving. Since all these collaborative actions are purely classical, they can therefore be implemented using a Classical SMPC. At the end, the clients decrypt their output if there was no abort.

As is apparent, the heart of the construction is the new DBQC Resource. It guarantees that, even when all but one client collude with the server, the data of the honest client and the computation remains hidden from the coalition. Since the VBQC client-encrypted state depends on these parameters, no coalition of malicious players can learn its description. This step does not require verifiability but only blindness, it is therefore possible to construct it from the simpler UBQC Protocol [21].

There again, we transform the step (i) of this UBQC Protocol into a multi-party version, which in the single-client version consists of the creation of states $|+\rangle$ rotated by a random angle $\theta(v)$ around the Z axis. For this we develop and use a collaborative single-qubit state preparation sub-protocol. For each vertex of the graph, each Client sends one randomly rotated $|+\rangle$ state and the Server performs successive gate-teleportations using these qubits. The effect of this procedure is to apply a collective encryption that depends on the rotation angles of all the Clients' states. The Clients send their rotation angles to the classical honest Orchestrator, who then knows the encryption and can effectively drive the rest of the UBQC Protocol. At the end it does not reveal the keys which would allow either the Clients or the Server to decrypt. This results in the computation and inputs of honest clients being blind to any coalition of malicious players.

To obtain the full protocol, the Orchestrator is replaced by a Classical SMPC which also chooses the unitary implemented by the DBQC Protocol so that it prepares the required VBQC client-encrypted state. The SMPC then drives the DBQC Protocol by computing the appropriate measurement angles. In a similar way, it drives the subsequent VBQC computation and checks that all trap measurements are correct, then telling the Clients whether they should accept the computation or abort. Note that the computation applied by the DBQC Protocol is constant-depth, independent of the size of the unitary computed later in the VBQC Protocol, thus making the preparation of the VBQC client-encrypted

states very efficient compared to the single-client version. This highly modular approach to protocol design makes improving the efficiency of our protocol simpler since any improvement in one of the sub-components immediately translates into an improvement on the global protocol so long as the conditions for composability are met.

Our protocol achieves a simultaneous optimization of the number of communication rounds, the Client-side memory size, and the operation complexity in a way that goes beyond the usual MPQC trade-off. More precisely, symmetric protocols such as [40] can be modified using gate teleportations into asymmetric versions where one Client plays the role of the Server. This allows to modify the topology of the network required to run the protocol as well as the number of rounds. Yet, in doing so, the complexity of the operations that need to be performed by the clients remains the same. They still need to have local fault-tolerant computation capabilities, whereas our scheme reduces this to single qubit gates. In addition, the overhead of our protocol with respect to the amount of qubits that need to be sent by each client compared to the single-client version is only a constant multiplicative factor of 9. Hence, our protocol is well adapted to the currently foreseen development of quantum computing services accessible through a quantum internet.

The second implication is an affirmative answer to the possibility of using the MBQC model to delegate MPQC in the dishonest majority setting. In fact, the inability to cope with client-server collusions in [76] outlined the essential role played by verification: it secures the key release in MBQC computations which could otherwise be vulnerable as the key depends on possibly corrupted qubits measured in the previous rounds. However, direct approaches trying to uplift the trap-based technique for verifying a delegated MBQC computation to the multi-client case based on letting each client trappify sub-graphs of the computation graph were not successful. This is due to the necessity to ensure that there always exists a computation path that performs the desired computation which in turn requires collaboration between the clients. Avoiding leaks about the location of the traps during these collaboration steps was a long-standing open question that we solve using DBQC to prepare non-verified states that would nonetheless be sufficient to verify the subsequent computation.

Sketch of Proof Techniques. The full protocol is the result of sequential composition of DBQC with VBQC and the Classical SMPC. The DBQC Protocol is proven secure in the AC framework, which involves showing that the protocol implementation cannot be distinguished from an ideal secure-by-design resource. This is done by exhibiting a Simulator that can be attached to the Ideal Resource for DBQC in order to make it indistinguishable from the real-world protocol in a statistically-secure sense. The technique used is similar to that of [41]: the Simulator sends half-EPR pairs to the Server. Any possible deviations from the protocol on these qubits can then be transferred via teleportations to the qubits of the Ideal Resource. Since these operations could have been performed by the malicious players on their own (the Simulator does not know the computation being performed), this also proves the blindness of the DBQC Protocol.

We prove that the VBQC Protocol is secure in the AC framework so long as the initial state has been prepared as a VBQC client-encrypted state. As DBQC provides blindness but not verifiability (and therefore not all components of the full protocol are verifiable), if we wish to compose these two protocols using the AC framework, we therefore need to show that the output state of the DBQC Protocol can be rewritten as such a state. This leads to defining the set of "good-enough" states, that

are correct VBQC client-encrypted state up to a deviation by the malicious parties that is independent on the inputs and parameters of the honest parties, and show that the DBQC Protocol produces exactly these states. The deviations that happen during the DBQC Protocol need to be commuted to the end of the protocol (an honest DBQC Protocol produces a non-deviated VBQC client-encrypted state). However, because this deviation models the action of the possibly dishonest coalition, it must remain independent of the secret parameters of the honest players. We prove that this additional condition is indeed satisfied for the DBQC Protocol, implying that, irrespective of the action of a dishonest coalition, the adversarial deviation during the non-verifiable DBQC Protocol can be commuted and pushed into the VBQC Protocol.

It is then possible to apply the security of the VBQC Protocol as defined in the AC framework and replace its execution with an Ideal Resource which either outputs the correct state at the end or aborts. The protocol obtained after these substitutions simply takes as input the input qubits of each Client and either successfully produces the correct output of the computation or aborts, which is trivially secure. The security of DBQC and VBQC being information theoretic, the proposed Delegated MPQC Protocol is an information-theoretic upgrade of the Classical SMPC.

Chapter Outline. An iterative construction of the protocol and high-level presentation of it is given in Section 6.2. Section 6.3 then details the construction of two double-blind state preparation protocols. Then we present the DBQC Protocol and prove its blindness. Section 6.4 proves that the DBQC Protocol can be used to bootstrap verifiability of the whole protocol. While we use these double-blind protocols to implement a multi-party VBQC client-encrypted state preparation, they are in fact more general and we believe they will find other applications in the future. Finally, the full protocol is presented in Section 6.5 along with its security statement and proof. As proof of the simplicity of our classical resource, Section 6.6 proposes a concrete construction for the Classical SPMC scheme that is used to implement the whole protocol.

Related Works. Table 6.1 below gives a comparison of our protocol with the peer-to-peer protocols of [40] and [90] and with the more recent semi-delegated protocol of [5]. Section 6.7 gives a more in-depth analysis, showing that our protocol closes gaps left as open questions in previous results. In the table, N is the number of parties, d the depth of the computation (MBQC for this work, circuit for [90] and $\{\mathsf{T}, \mathsf{CNOT}\}$-depth for [40]), g the number of gates in the computed circuit, t the number of $\mathsf{Z}(\pi/4)$ gates, c the number of CNOT gates, C_{dist} the code distance used in [90] and η a statistical security parameter. Q stands for quantum and C for classical. Note that we list below the optimal parameters for the protocol's execution. In particular the network topology of [40] and [90] can be star-shaped as well with one player acting as a router (they only provided unanimous abort anyway), but this would degrade their performance in terms of quantum communication rounds. In that sense the Complete Graph is more natural for these protocols.

6.2 High-Level Construction of a Delegated MPQC Protocol from VBQC

The main purpose of this chapter is to build a Protocol constructing a Multi-Party Quantum Computation Resource in a way that is composably secure, that can be delegated by relatively powerless Clients to a

Metric	[40]	[90]	[5]	This work
Type	Stat. upgrade of CSMPC	Statistical	Comp. (FHE + CSMPC)	Stat. upgrade of CSMPC
Abort	Unanimous	Unanimous	Identifiable	Unanimous
Composability	Composable	Stand-Alone	Stand-Alone	Composable
Max Malicious Players	$N-1$	$\lfloor \frac{C_{dist}-1}{2} \rfloor$	$N-1$	$N-1$
Protocol Nature	Symmetric	Symmetric	Semi-Delegated	Delegated
Network Topology	Q and C: Complete	Q and C: Complete	Q and C: Complete	Q: Star / C: Complete
Q Operations	F.T. Q. Comp	FT Q Comp	FT Q Comp	Cl.: Single Qubit Serv.: FT Q Comp
Classical SMPC	Clifford Computation, Operations in $\mathbb{Z}_2$, CT	CT	Clifford Computation, FHE verification	Operations in $\mathbb{Z}_8$, $\mathbb{Z}_2$, CT
Rounds (C or CSMPC)	$\mathcal{O}(g+\eta(N+t))$	$d+2$	$\mathcal{O}(1)$	$d+5$
Rounds (Q)	Par.: $\mathcal{O}(Nd)$ Seq.: $\mathcal{O}(N(N+t+c))$	Par.: 3 (2 if C output) Seq.: $\mathcal{O}(\eta^2(N+t))$	Par.: $\mathcal{O}(N^4)$	Par.: 2 (1 if C output) Seq.: $\mathcal{O}(\eta N d)$
Size of Q Memory	Par.: $\mathcal{O}(\eta^2(N+t)))$ Seq.: $\mathcal{O}(\eta^2 N)$	Par.: $\mathcal{O}(\eta^2 N(N+t))$ Seq.: $\mathcal{O}(N^2)$	Par.: $\mathcal{O}(tN^9\eta^2)$	Cl.: 3 (0 if C I&O) Serv. (par.): $\mathcal{O}(\eta N^2 d)$ Serv. (seq.): $\mathcal{O}(\eta N d)$

Table 6.1: Comparison with [40, 90, 5]. Q stands for quantum and C for classical. Cl. and Serv. stand for Client and Server respectively. Stat. means statistical, FT stands for Fault-Tolerant and CT for Coin-Toss. Seq. and Par. refer to whether the protocol is optimised for low qubit memory, therefore performing all quantum communications sequentially, or trying to parallelise the communication rounds at the expense of manipulating a larger number of qubits.

.

powerful quantum Server, that tolerates Client-Server collusion and requires only a single honest Client to run securely. We tackle this by first deconstructing and analysing the single-client VBQC Protocol of [78]. We determine the steps which need to be updated to transform it into a multi-client setting, together with the conditions that these replacement steps need to satisfy. Having defined these, we construct a Delegated Multi-Party Quantum Computation Protocol (DMPQC) by making use of the elegant composability property of the AC framework. We show later in Section 6.3 how to instantiate these new multi-party subroutines. The formal presentation of the DMPQC Protocol and associated security guarantees can be found in Section 6.5.

As we will manipulate quantum states ρ involving many qubits at different locations and from different origins, it will be convenient to label the reduced state at given location indexed by v and being provided by Client j as $\rho_j(v)$. By extension, index j might be replaced by a set of clients indices (typically H for honest clients and M for malicious ones). We will apply the same notation for classical variables denoting secret parameters at position v and for Client j, such as $\theta_j(v)$ and $r_j(v)$. When a state or classical variable depends collectively on all Clients, the subscript j will usually be dropped.

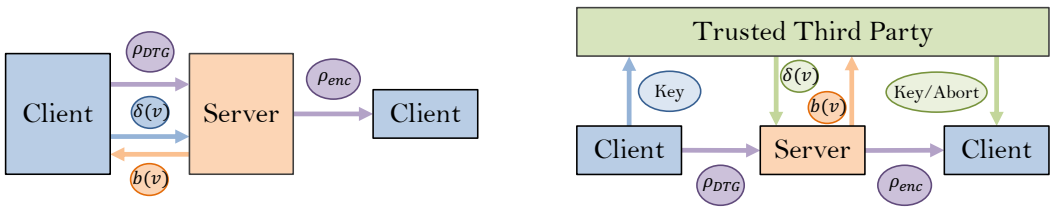
We assume that an Authenticated Classical Channel is available between any two parties, a Secure Classical Channel is available between all Clients and the Orchestrator, and an Insecure Quantum Channel is available between all Clients and the Server.

6.2.1 Deconstructing the VBQC Protocol

The protocol for Multi-Party Quantum Computation will rely on the VBQC construction. The single-client VBQC protocol of [78] can be decomposed in the following parts: choosing secret parameters (graph colouring, encryption keys); preparing the *VBQC client-encrypted state* (comprising encrypted quantum inputs, dummies and encrypted $|+\rangle$ states sent by the Client to the Server during the initialization step of Protocol 5); sending single qubits; applying entangling operations and classically-driven measurement; receiving quantum outputs; aborting or decrypting. Transforming this protocol into a multi-party one

requires modifying each of these individual steps so that they can be performed collectively by several clients without compromising the blindness, even in the situation where some Clients and the Server collude.

First, we remark that the protocol is perfectly equivalent to one where a Trusted Third Party receives from the Client the classical parameters (chosen at random within their allowed range) that are used when preparing the corresponding states and encrypted inputs to the Server. Second, the knowledge of these classical parameters and of the unitary U to apply is indeed sufficient for the Trusted Third Party to drive the VBQC computation and also verify that the traps have been measured correctly, outputting either the keys to the Client in case of success or instructing it to abort if any trap failed. In this modified setup, the only operations that the Client would still need to perform are encrypting its inputs and preparing dummies and encrypted $|+\rangle$ states to generate the VBQC client-encrypted state, sending the encryption key to the Trusted Third Party, sending the state to the Server and, if there is no abort, recovering the output state from the Server, the keys from the Trusted Third Party and decrypting to get the final state. This change is pictured in Figure 6.1.



(a) Simple VBQC Outline. (b) VBQC with Trusted Third Party.

Figure 6.1: Replacing the classical steps of the Client in the VBQC Protocol with a Trusted Third Party. The Client (C), Server (S) and Trusted Third Party are represented in blue, orange and green respectively. Their classical communications are represented in the same colour while quantum communications are in purple. The state ρ_{DTG} is a VBQC client-encrypted state with the blue *Key* being the classical description of this state while ρ_{enc} is the state returned by the Server to the Client at the end of the computation associated with the measurement angles δ and results b, encrypted with the green Q-OTP *Key*. The Client decrypts this state if if has not received Abort from the Trusted Third Party.

6.2.2 Reconstructing a DMPQC Protocol with the DBQC Ideal Resource

Using the remarks above, we conclude that the VBQC Protocol can be driven by a Trusted Third Party independent of the Client – or equivalently by a Classical SMPC in the context of multiple Clients – provided that the Server receives a state that corresponds to the VBQC client-encrypted state. In that case, the verifiability of the VBQC protocol by itself guarantees that the probability a party cheated without the honest parties noticing is negligible. When following this path, two steps need attention: we need to (i) find a way to collaboratively prepare the VBQC client-encrypted state; and (ii) specify the procedure performed by each Client after receiving the quantum output from the Server.

We start with the easier case of outputs. Indeed, the only point that needs to be specifically addressed is the verification of the trap qubits placed in the output layer before receiving the keys and decrypting. These traps need to be measured and pass the test. But in doing so, the Server should not learn which

of the output layer qubits of the honest Clients are computational ones. The solution is to first send the qubits of the output layer to the Clients – each Client receiving the qubits corresponding to the base-locations of its own output –, then having the Classical SMPC reveal to each Client where the traps are among its qubits, and have them measured. The Classical SMPC then verifies that all the traps are correct which guarantees with overwhelming probability that the Server has sent to each (honest) Client their intended quantum states of the output layer, and in particular their quantum output. Hence, under the condition that this verification passes, the Classical SMPC can safely send the decryption keys for each Client individually as the malicious coalition cannot access the quantum states of the honest Clients. The Clients can then decrypt their outputs.

Regarding the inputs, if the Clients had access to a *VBQC Client-Encrypted State Preparation Ideal Resource* that would securely provide the Server with the single-client VBQC client-encrypted state and provide the Classical SMPC with the secret parameters of this state, this would solve the problem entirely. The Classical SMPC would just have to instruct the Server according to the VBQC protocol. Unfortunately, such Ideal Resource would be hard to construct as it would require to find a protocol that is already performing some form of MPQC – albeit a simple one. The problem arises from the fact that each Client should in that case be able to verify that the output state of the protocol implementing this State Preparation Ideal Resource is indeed a correctly generated VBQC client-encrypted state (due to the fact that an MPQC Resource is usually defined so that it either produces the correct output or aborts).

One way around this issue is to ask the resource to produce a correct VBQC client-encrypted state up to a deviation chosen by the Server (independent on any of the secret parameters), while maintaining the blindness about the state prepared, ie. the position, type and encryption of each qubit. For this purpose we introduce a new Resource called Double-Blind Quantum Computation (DBQC). This would be sufficient for our purpose as the deviation on the VBQC client-encrypted state could then be treated as a deviation by the malicious parties during the execution of the VBQC Protocol.

Double Blind Quantum Computation Ideal Resource. Resource 20 allows N Clients to submit their part of a collectively possessed input quantum state ρ_{inp} and an Orchestrator to input a classical description of a unitary transformation U to apply to ρ_{inp}. A coalition of malicious parties may induce a deviation from the unitary specified by the Orchestrator by way of a CPTP map $\mathcal{E}$, applied to the input state and adversarially-chosen ancillary quantum state instead of the legitimate transformation. In both cases, the Server receives from the Resource the output quantum state which is encrypted by a Q-OTP while the Orchestrator gets the corresponding randomly chosen key k, which guarantees the blindness of the scheme.

While this resource does not prevent tampering, (i.e. a coalition could get an output different than the expected $\mathsf{QOTP}_k \circ U(\rho_{C_1,\ldots,C_N})$), it is blind since no choice of $\mathcal{E}$ will let the coalition discover inputs or outputs beyond the permitted leakage, as it always gets an encrypted copy of it without the description key k. The permitted leakage represents everything a coalition can learn aside from the input states of the malicious Clients. In our protocol, the permitted leakage will be the computation graph G used to implement U. When G is a universal graph for MBQC, this amounts to leaking its size, or equivalently an upper-bound on the number of gates used to implement the computation chosen by the Orchestrator in the circuit model.

Resource 20 Double-Blind Quantum Computation

Inputs:

- Each Honest Client $j \in [N]$ has a quantum register $\mathcal{X}_j$ which contains their respective part of a collectively possessed state ρ_{inp}.
- The Orchestrator inputs the classical description of a unitary U.
- The Server has no honest input.
- All Clients and the Server have filtered interfaces controlled by $\{c_j\}_{j\in[N]\cup\{S\}}$ (set to 0 in the honest case). When a coalition cheats, they collectively send quantum state ρ_M and the classical description of a CPTP map $\mathcal{E}$.

Computation by the Resource:

1. It samples uniformly at random a Q-OTP key $k = (k_X, k_Z)$ of the size of the output state.
2. If all $c_j = 0$, the state $\mathsf{QOTP}_k \circ U(\rho_{inp})$ is produced at the Server's interface.
3. If there are parties j with $c_j = 1$, the Ideal Resource first sends them the leakage l_ρ. Then, the malicious parties send their additional inputs to the Resource. The state $\mathsf{QOTP}_k \circ \mathcal{E}(\rho_{H,M,U})$ is produced at the Server's interface where $\rho_{H,M,U}$ includes the inputs of the honest Clients (H), the malicious parties (M), and the Orchestrator's description of U.
4. The Q-OTP key $k = (k_X, k_Z)$ is output at the Orchestrator's interface.

The production of a correct VBQC client-encrypted state up to a deviation can be then accomplished by using this new DBQC Ideal Resource. The input of the Clients to this Ideal Resource are as follows. One Client will be assigned to each base-location of the Dotted-Triple Graph used in the VBQC Protocol (how this choice is performed is described later) and supply rotated $|+_\theta\rangle$ states for these positions (three for primary locations and nine for added locations). Naturally, the input base-locations are assigned to the Client who is supposed to provide the associated input in the VBQC computation. For these, the Client chooses one position at random to send its encrypted input qubit, and for the other two sends rotated $|+_\theta\rangle$ states. It communicates the position of its input to the Classical SMPC, which then chooses a colouring of the Dotted-Triple Graph that satisfies these positioning requirements. The Classical SMPC (acting as the Orchestrator) can choose a Clifford unitary U which turns some of the $|+_\theta\rangle$ states into dummies and applies identity on the others, keeping them for the traps and computations. All this can be done using a unitary U made of SWAP, H and I gates. Section 6.3 gives an explicit DBQC Protocol for implementing the DBQC Ideal Resource and it will be proved in Section 6.4 that, by carefully choosing the measurement pattern of DBQC, the output state can be forced to be any VBQC client-encrypted state up to a deviation by the Server that does not depend on the secret parameters of the state. This DBQC Protocol will consist of two phases: a state preparation step – during which each Client sends both the qubits that will become part of the VBQC Dotted-Triple Graph and auxiliary qubits that are only used during the DBQC Protocol – and a computation step corresponding to an execution of UBQC on the collaboratively-generated state resulting from this state-preparation. The MPQC Protocol resulting from these modifications is represented Figure 6.2.

We need to make a final modification to the VBQC Protocol which is motivated by our security analysis from Section 6.4.2. We assume that the MBQC base graph used to perform the joint unitary has degree 1 for all input qubits.[1] Then, after the entanglement operation on the DTG has been performed by the Server, it starts by applying an encrypted bridge operation (instructed by the Classical SMPC)

[1] The universal brickwork state satisfies this property, meaning that our assumption does not restrict the computations that can be performed using our scheme.

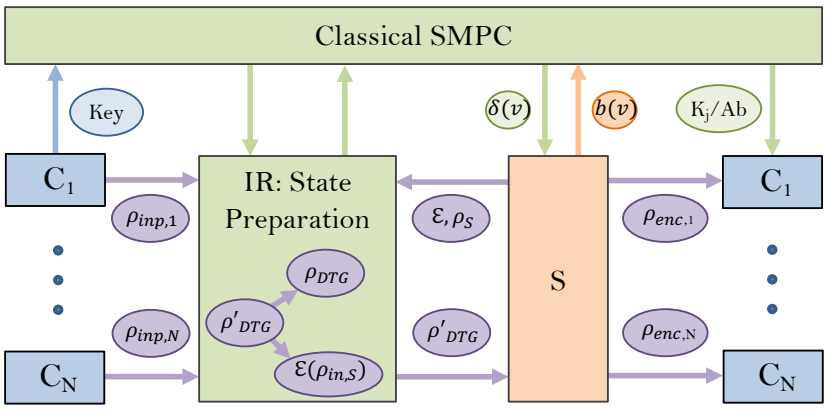


Figure 6.2: Replacing the classical steps of the single Client in VBQC with a Classical SMPC and the VBQC client-encrypted state preparation with a possibly-deviated multi-party state preparation with the participation of Clients C_i. The state ρ'_{DTG} corresponds either to the honestly prepared VBQC client-encrypted state if there was no deviation, or a deviated state $\mathcal{E}(\rho_{in,S})$ where $\rho_{in,S}$ corresponds to a state containing the honest Client's input and VBQC state qubits and an auxiliary state supplied by the malicious coalition. At the end, each Client C_i receives its part of the encrypted output state and its associated Q-OTP key K_i (unless there was an abort Ab).

on the added vertices corresponding to the inputs' only edge.

To summarise, the Delegated MPQC Protocol consists of the following steps:

1. The Clients send qubits for the state preparation. These include qubits that will be part of the DTG and qubits for realising the DBQC Protocol, all are either their encrypted input qubits or rotated qubits $|+_\theta\rangle$. The inputs are placed by each Client uniformly at random among the qubits of their associated base-location. The corresponding secret parameters are sent to the Classical SMPC.
2. The Server performs the rest of the DBQC Protocol with the Classical SMPC, which instructs it to measure some qubits in a rotated basis that depends on the secret parameters above. The aim of this DBQC Protocol is to transform some of the encrypted $|+\rangle$ states into dummies while keeping the rest of the state as is. The Server returns the measurement results from the DBQC Protocol to the Classical SMPC. At the end of this step, the Server is in possession of a VBQC client-encrypted state whose parameters are known to the Classical SMPC (up to a final deviation).
3. The Classical SMPC instructs the Server to perform the VBQC Protocol performing the Clients' computation on this prepared state, with the only difference being that the first step after entangling the DTG graph state consists of encrypted bridge operations for computation qubits for input edges (as described in Section 2.3.1). At the end, the Server returns to each Client all the qubits in their assigned output base location.
4. The Clients and the Classical SMPC perform a key-release step, at the end of which they either abort or receive the decryption keys for their output.

6.2.3 Usage of the Classical SMPC Ideal Resource

We have seen above that the Classical SMPC allows the Clients to perform collaborative tasks in a secure way. We detail below how it mediates the interactions between the Clients themselves and between the Clients and the Server.

Initialisation SMPC. This first call to the SMPC prepares its own internal state (there is no output at this stage): it receives from each Client a classical description of the secret parameters of all qubits that it has sent to the Server along with the location of its input among them. Some of those qubits correspond to the ones that will be used as inputs to produce the VBQC client-encrypted state upon which the actual computation will take place, while others are used to perform the DBQC Protocol which will either apply a Hadamard on qubits that are destined to become dummies or Identity to the rest (so that the combined qubits form a Dotted-Triple Graph). The Classical SMPC chooses the colouring of the Dotted-Triple Graph associated with the base graph (the one computing the unitary chosen by the Clients). This colouring must be compatible with the positions of the Clients' inputs (i.e. the computation qubit in the input base-locations is determined by each Client's choice). The Classical SMPC then defines the unitary applied through the DBQC Protocol: the colouring deterministically defines at which positions to apply the H and I operations.

Trusted Orchestrator SMPC. The second execution of the Classical SMPC is called during the DBQC Protocol used to apply these operations. Protocol 18 requires an honest Orchestrator for it to construct the DBQC Resource 20. Since there is no such party in the actual DMPQC Protocol below, the aim of this call is to implement this trusted party. During its first call, the Classical SMPC has stored in its internal state the transformation that should be applied (the location of the H and I gates required to turn some qubits into dummies while leaving the others unaffected). It then transforms it into an MBQC computation. It receives classical information from the Server during the state preparation phase of the DBQC Protocol (corresponding to measurement results) and, knowing the state of the qubits sent by the Clients for this state preparation, deduces the state of the qubits in the computation graph at the end of this step. It can then instruct the Server with the measurement angles defined by the DBQC Protocol as an honest Orchestrator would. Since the unitary that is being applied consists only of Clifford operations, the SMPC can send all the measurement angles at once, the corrections will simply translate into a different Q-OTP key on the final state. The Server returns the measurement outcomes of all measured qubits to the Classical SMPC.

Classical Instructions SMPC. Finally, since it knows all the parameters of the Dotted-Triple Graph, it can also drive the VBQC computation on the original input qubits by instructing the Server to measure the qubits in a certain angle. For the last layer, it tells to each Client where its computation, trap and dummy qubits are. It sends them the measurement angle for the traps and recovers the measurement outcomes. It knows internally the correct value of the traps and so can verify that they have been all correctly measured (the ones measured by the Server during the computation and the ones now measured by the Clients). If the verification passes, it sends to each Client the keys for decrypting its output.

6.3 Double-Blind Resources for Collaborative State Generation and Computation

We present in this section three quantum protocols that are *double-blind* in the sense that neither the Clients nor the Server (nor any incomplete coalition) have any knowledge about a given property of the state produced by their executions. The first two correspond to state preparation protocols whose aim is to collaboratively prepare states from a given set without revealing which one has been effectively produced. More precisely, the first protocol produces rotated states from the set $\{|+_\theta\rangle\}_{\theta\in\Theta}$ while the second one produces BB84 state, i.e. from the set $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. We do not show the security formally of these two protocols as they are intended to be used as sub-routines. We however presented informally reasons why they reveal no information about the prepared states. The third protocol is the Double-Blind Quantum Computation Protocol, which uses the Double-Blind Rotated State Preparation Protocol on top of the UBQC Protocol to collaboratively perform a computation without revealing which unitary has been applied. The security of this protocol is proven as the emulation of the DBQC Resource 20.

6.3.1 Double-Blind Rotated State Preparation

As mentioned above, the DBQC Protocol will apply the UBQC Protocol on a state that has been collaboratively encrypted. In order to satisfy the blindness condition against coalitions of the Server and up to $N-1$ Clients, the Double-Blind Rotated State Preparation (Protocol 16) must ensure that no coalition of less than N Clients has enough information about the encryption keys of the qubits used in the UBQC Protocol. This imposes that all the Clients contribute to the encryption, each sending at least one qubit per non-output location of the UBQC computation graph G. This way, each qubit used in the UBQC Protocol always contains random bits from at least one honest Client. In the Rotated State Preparation Protocol, one Client provides an encrypted input and all others send rotated $|+_\theta\rangle$ states. Upon receiving the qubits of each Client, the Server applies a series of CNOT's between the received qubits followed by a computational basis measurement of all but one qubit. By properly choosing the control and targets of the CNOT's this results in a single qubit whose encryption depends on the secret parameters of all the Clients. This can then be used to perform the UBQC computation. The formal version of the Double-Blind Rotated State Preparation is given in Protocol 16.

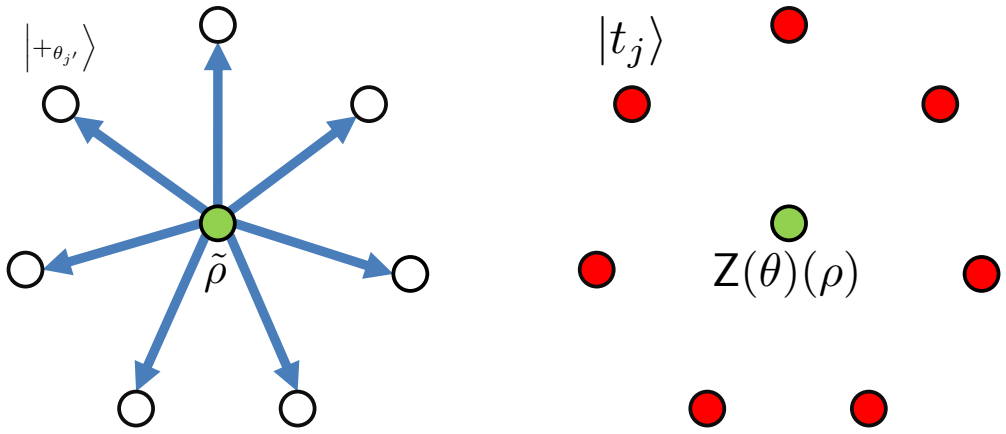
Protocol 16 Double-Blind Rotated State Preparation

Input: Client j has as input a quantum register containing a single qubit. The other parties have no input.

Protocol:

- Client j chooses $\theta_j \in_R \Theta$ uniformly at random, applies $\mathsf{Z}(\theta_j)$ to its input and send it to the Server. It keeps the value θ_j as output.
- Client $j' \neq j$ chooses $\theta_{j'} \in_R \Theta$ uniformly at random, prepares $\left|+_{\theta_{j'}}\right\rangle$ and send it to the Server. It keeps the value $\theta_{j'}$ as output.
- For each $j' \neq j$, the Server applies $\mathsf{CNOT}_{j,j'}$ between the qubits received from Clients j and j', with the first being the control and the second the target. It measures the target qubit (sent by Client j') in the computational basis with measurement outcome $t_{j'}$. It keeps as output the vector t containing all the measurement outcomes and the register containing the qubit of Client j.

This Double-Blind Rotated State Preparation is represented eight Clients in Figure 6.3.



(a) The Server receives the qubits and applies CNOT gates (the central qubit is the control, the rest are targets).

(b) The Server measures all qubits but the central one in the computational basis and gets outcomes $t_{j'} \in \{0, 1\}$.

Figure 6.3: Double-Blind Rotated State Preparation for eight Clients. Client j supplies the central input qubit in state $\tilde{\rho} = \mathsf{Z}(\theta_j)(\rho)$, every other Client j' sends $\left|+_{\theta_{j'}(v)}\right\rangle$.

Lemma 6.1 (Result of Double-Blind Rotated State Preparation)**.** *After an honest execution of Protocol 16 with input state ρ, the state of the unmeasured qubit held by the Server is $\mathsf{Z}(\theta)(\rho)$:*

$$\theta = \theta_j + \sum_{j' \neq j} (-1)^{t_{j'}} \theta_{j'} \tag{6.1}$$

Proof. It is sufficient to prove the lemma for a pure input state $|\phi\rangle = \alpha |0\rangle + \beta |1\rangle$. Let $|+_\theta\rangle$ with $\theta \in \Theta$ be a rotated quantum state. Then, we apply a CNOT gate with the input state as control and the rotated state as target, followed by a measurement of the second qubit in the computational basis. Let $t \in \{0, 1\}$ be the measurement result. After tracing out the second qubit post-measurement, the system is in the following state:

$$\begin{aligned} \langle 0|_2 \mathsf{X}_2^t \mathsf{CNOT}_{12} |\phi\rangle |+_\theta\rangle &= \langle 0|_2 \mathsf{X}_2^t \left(\alpha |00\rangle + \alpha e^{i\theta} |01\rangle + \beta |11\rangle + \beta e^{i\theta} |10\rangle\right) \\ &= \langle 0|_2 \left(\alpha |0\rangle + \beta e^{i\theta} |1\rangle\right) |t\rangle + e^{i\theta} \langle 0|_2 \left(\alpha |0\rangle + \beta e^{-i\theta} |1\rangle\right) |t \oplus 1\rangle \\ &= \mathsf{Z}(\theta) |\phi\rangle \langle 0|t\rangle + \mathsf{Z}(-\theta) |\phi\rangle \langle 0|t \oplus 1\rangle \end{aligned} \tag{6.2}$$

Therefore, the result of this single step is $\mathsf{Z}((-1)^t \theta) |\phi\rangle$. Recall that Client j performs an encryption on its input using operation $\mathsf{Z}(\theta_j)$. Replacing the result above in the sequence of CNOT's and measurements performed by the Server where the control is the qubit sent by Client j and the target qubits are those sent by the Clients $j' \neq j$ yields the desired result.

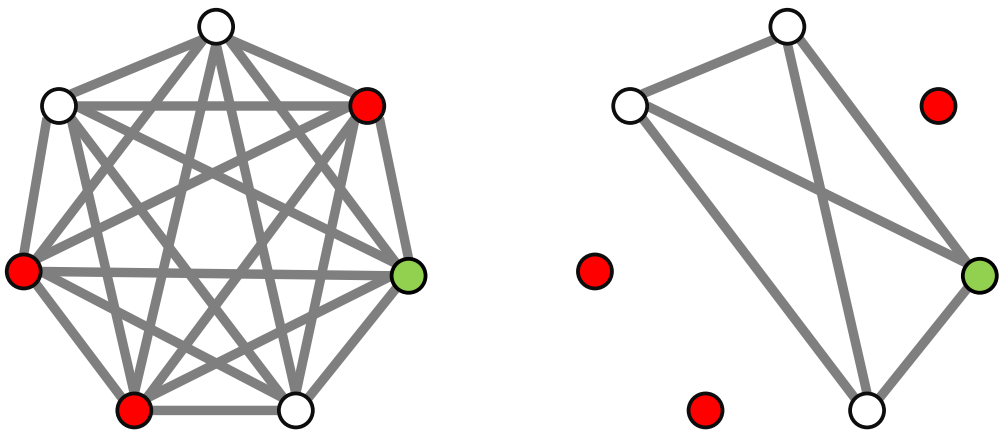
■

Note that the state after this protocol has been executed could have been obtained by having each Client send its inputs to a quantum-enabled Orchestrator and let it perform the state rotation by θ along the Z axis. What is important here, is that the computed angles θ from equation 6.1 will appear random to any party as long as at least a single unknown angle. This property will provide the security of our DBQC Protocol against collusions of up to $N-1$ Clients.

6.3.2 Double-Blind BB84 State Preparation

We present here a a novel way of preparing a state from the BB84 set collaboratively such that no incomplete coalition of parties has any information about the final state. Note that while the rest of this Chapter does not make use of this subroutine, future collaborative MBQC-based protocols may find it useful due to the interesting properties of states from this set when inserted in vertices of graph states. This would allow for performing blind bridge or break operations, potentially moulding the computation graph without revealing how.

This double-blind protocol works as follows. Each of the N Clients is instructed by its input to prepare either a state in $\{|0\rangle, |1\rangle\}$ or $\{|+\rangle, |-\rangle\}$ and send it to the Server. The Server prepares a qubit in the state $|+\rangle$ and applies CZ to each pair of qubits. The result is presented in Figure 6.4 for 6 Clients. The Server then measures each of the qubits sent by Clients in the $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ basis and keep its own (unmeasured) qubit as output. The formal description of the collaborative Double-Blind BB84 State Preparation is given in Protocol 17.



(a) Entanglement of Client and Server qubits according to a complete graph.

(b) Dummies are unentangled from the rest of the state. All qubits but the green one are measured in the $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ basis.

Figure 6.4: Example of possible entanglement configuration with $N = 6$ Clients. Red vertices are dummies, white vertices are qubits in a state from $\{|+\rangle, |-\rangle\}$, the green vertex is the Server's qubit in state $|+\rangle$.

We now describe the state of the unmeasured qubit at the end of the protocol in Lemma 6.2, proving that if all player's acted honestly this register contains a qubit in a state from $\{|+\rangle, |-\rangle\}$ in the case

Protocol 17 Double-Blind BB84 State Preparation.

Inputs: Each Client $j \in [N]$ has as input $a_j \in \{0,1\}$
The Protocol:

1. Each Client j samples a bit uniformly at random $b_j \in_R \{0,1\}$. It prepares and sends to the Server a qubit in the state $|\psi_j\rangle = \mathsf{H}^{a_j} |b_j\rangle$. It then sets b_j as its output.
2. The Server prepares a qubit in the state $|+\rangle$.
3. The Server entangles the qubits by applying a CZ gate between each pair of qubits in its possession (forming a complete graph state across all $N+1$ qubits).
4. The Server measures all qubits sent by the Clients in the $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ basis. Let c_j be the result of the measurement on the qubit sent by Client j and $\mathbf{c}$ the corresponding string. It sets the string $\mathbf{c}$ and the unmeasured qubit as its output.

where $\bigoplus_{j=1}^{N} a_j = 0$, and from $\{|0\rangle, |1\rangle\}$ otherwise.

Lemma 6.2 (Result of Double-Blind BB84 State Preparation)**.** *Let $D = \{j \in [N] \mid a_j = 0\}$ be the set of indices of qubits in the computational basis and $R = D^c = [N] \setminus D$ the set of indices of qubits in states $\{|+\rangle, |-\rangle\}$ sent by the Clients. Let $n = \#R$, $d = \sum_{j\in D} b_j$, $e_j = b_j + c_j + d$ and $e = \sum_{j\in R} e_j$ with $\mathbf{e}$ the corresponding string. The unmeasured qubit (indexed O) at the end of an honest execution of Protocol 17 is in the state $|\phi_{n,d,\mathbf{e}}\rangle = \mathsf{Z}^d \mathsf{H}^n \mathsf{Z}^{e+\left\lfloor \frac{n}{2} \right\rfloor} |+\rangle$.*

Proof. Since we analyse correctness, we can suppose without loss of generality that the first n qubits are in the state $|+\rangle$ or $|-\rangle$ (such that $R = [n]$ and $D = [N] \setminus [n]$). The state the N qubits sent by the Clients is $|\psi_j\rangle = \mathsf{Z}^{b_j} |+\rangle$ for $j \in R$ and $|\psi_j\rangle = |b_j\rangle$ for $j \in D$. The state before the entangling operation performed by the Server is therefore (where the lower index on unitaries indicating to which qubit they are applied):

$$\bigotimes_{j\in R} \mathsf{Z}_j^{b_j} |+\rangle \bigotimes_{j\in D} |b_j\rangle = \prod_{j\in R} \mathsf{Z}_j^{b_j} \bigotimes_{j\in R} |+\rangle \bigotimes_{j\in D} |b_j\rangle \tag{6.3}$$

The entangling operations between qubits from set D only add a global phase and can be disregarded. Let $R^* = R \cup \{O\}$ (the last one being the Server's qubit). The entangling operation between qubits $i \in D$ and $j \in R^*$ is equivalent to applying $\mathsf{Z}_j^{b_i}$. Since this is done on all qubits $j \in R^*$ for each qubit $i \in D$, this applies $\prod_{i\in D} \left(\prod_{j\in R^*} \mathsf{Z}_j\right)^{b_i} = \left(\prod_{j\in R^*} \mathsf{Z}_j\right)^{\sum_{i\in D} b_i} = \mathsf{Z}_O^d \prod_{j\in R} \mathsf{Z}_j^d$. This operation commutes with the other CZ gates and the resulting state on the Server's side before it performs the measurements is therefore:

$$\prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} \prod_{j\in R} \mathsf{Z}_j^d \prod_{j\in R} \mathsf{Z}_j^{b_j} \bigotimes_{j\in R} |+\rangle \bigotimes_{j\in D} |b_j\rangle \otimes \mathsf{Z}_O^d |+\rangle = \prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} \prod_{j\in R} \mathsf{Z}_j^{b_j+d} \bigotimes_{j\in R} |+\rangle \bigotimes_{j\in D} |b_j\rangle \otimes \mathsf{Z}_O^d |+\rangle \tag{6.4}$$

The qubits in the computational basis are unentangled from the rest of the state and so measuring them has no influence on the output qubit at the end of the protocol. We can therefore trace them out, giving us the following reduced state:

$$\prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} \prod_{j\in R} \mathsf{Z}_j^{b_j+d} \mathsf{Z}_O^d |+\rangle^{\otimes n+1} \tag{6.5}$$

For $j \in R$, let c_j be the measurement result on the qubit j. The state of the measured qubits is then $\prod_{j\in R} \mathsf{Z}_j^{c_j} |+\rangle^{\otimes n}$. The probability of obtaining measurement result c_j on qubit j is $1/2$ independently of the other measurements so the probability of obtaining a given string of measurement $\mathbf{c}$ is $1/2^n$. Defining $e_j := c_j + b_j + d$, $\mathbf{e}$ the list of all values of e_j and $e = \sum_{j\in R} e_j$ and using the formula for the post-measurement state $|\psi_m\rangle = \frac{P_m|\phi\rangle}{\sqrt{p(m)}}$ for measurement result m, projector P_m and probability $p(m)$, the state of the unmeasured (output) qubit after projection is given by (with the measured qubits being traced out):

$$\begin{aligned} |\phi_{n,d,\mathbf{e}}\rangle :=& 2^{\frac{n}{2}} \langle +|^{\otimes n} \prod_{j\in R} \mathsf{Z}_j^{c_j} \prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} \prod_{j\in R} \mathsf{Z}_j^{b_j+d} \mathsf{Z}_O^d |+\rangle^{\otimes n+1} \\ =& 2^{\frac{n}{2}} \langle +|^{\otimes n} \prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} \prod_{j\in R} \mathsf{Z}_j^{e_j} \mathsf{Z}_O^d |+\rangle^{\otimes n+1} \end{aligned} \tag{6.6}$$

We now use the fact that $|+\rangle^{\otimes n} = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle$ to rewrite this state as follows:

$$|\phi_{n,d,\mathbf{e}}\rangle = \frac{2^{\frac{n}{2}}}{2^{n+\frac{1}{2}}} \Big(\sum_{\tilde{x}=0}^{2^n-1} \langle \tilde{x}| \Big) \prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} \prod_{j\in R} \mathsf{Z}_j^{e_j} \Big(\mathsf{Z}_O^d \sum_{x=0}^{2^n-1} |x\rangle |0\rangle_O + |x\rangle |1\rangle_O \Big) \tag{6.7}$$

Recall that $\mathbf{e} \cdot x$ designates the scalar product of $\mathbf{e}$ and x in $\mathbb{Z}_2^n$, $\mathrm{w_H}(x)$ is the Hamming weight of x and $\binom{n}{k}$ is the number of k-combination of a set of size n. We now use the following three identities to simplify the equation above:[2]

$$\begin{aligned} \prod_{j\in R} \mathsf{Z}_j^{e_j} |x\rangle =& (-1)^{\mathbf{e}\cdot x} |x\rangle \\ \prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} |x\rangle |0\rangle =& (-1)^{\binom{\mathrm{w_H}(x)}{2}} |x\rangle |0\rangle \\ \prod_{\substack{i,j\in R^* \\ i>j}} \mathsf{CZ}_{ij} |x\rangle |1\rangle =& (-1)^{\binom{\mathrm{w_H}(x)}{2}+\mathrm{w_H}(x)} |x\rangle |1\rangle \end{aligned} \tag{6.8}$$

We then get (simplifying the normalisation factor along the way):

$$|\phi_{n,d,\mathbf{e}}\rangle = \frac{1}{2^{\frac{n+1}{2}}} \sum_{\tilde{x}=0}^{2^n-1} \sum_{x=0}^{2^n-1} (-1)^{\mathbf{e}\cdot x + \binom{\mathrm{w_H}(x)}{2}} \langle \tilde{x}| \mathsf{Z}_O^d \Big(|x\rangle |0\rangle_O + (-1)^{\mathrm{w_H}(x)} |x\rangle |1\rangle_O \Big) \tag{6.9}$$

Using $\langle \tilde{x}|x\rangle = \delta_{\tilde{x},x}$ where δ is the Kronecker delta (dropping the index on the last remaining Z as there is no ambiguity any more):

$$|\phi_{n,d,\mathbf{e}}\rangle = \frac{\mathsf{Z}^d}{2^{\frac{n+1}{2}}} \sum_{x=0}^{2^n-1} (-1)^{\mathbf{e}\cdot x + \binom{\mathrm{w_H}(x)}{2}} \Big(|0\rangle + (-1)^{\mathrm{w_H}(x)} |1\rangle \Big) \tag{6.10}$$

[2]The first one stems from the fact that a -1 is applied if and only if $e_j = 1$ and $x_j = 1$, while the other two correspond to the fact that a -1 is applied by CZ if both bits of x upon which the gate acts are equal to 1

For $n = 0$ and $e = \varepsilon$ (the empty word), $|\phi_{0,d,\epsilon}\rangle = \mathsf{Z}^d |+\rangle$ (there is only the Server's qubit and all the rest are dummies). If $n > 0$, we can write $x = x'x_n$ where $x' \in \{0,1\}^{n-1}$[3] and $x_n \in \{0,1\}$ is the least significant bit of x (and similarly $\mathbf{e} = \mathbf{e}'e_n$):

$$|\phi_{n,d,\mathbf{e}}\rangle = \frac{1}{\sqrt{2}} \times \frac{\mathsf{Z}^d}{2^{\frac{n}{2}}} \sum_{x'=0}^{2^{n-1}-1} (-1)^{\mathbf{e}'\cdot x' + \binom{\mathrm{w_H}(x')}{2}} \Big(|0\rangle + (-1)^{\mathrm{w_H}(x')} |1\rangle\Big) + (-1)^{\mathbf{e}'\cdot x' + e_n + \binom{\mathrm{w_H}(x')+1}{2}} \Big(|0\rangle + (-1)^{\mathrm{w_H}(x')+1} |1\rangle\Big) \tag{6.11}$$

The term for $x_n = 0$ (on the upper line of the equation above) is exactly $\frac{1}{\sqrt{2}} |\phi_{n-1,d,\mathbf{e}'}\rangle$. For the term corresponding to $x_n = 1$ (on the lower line) we use the fact that $\binom{\mathrm{w_H}(x')+1}{2} = \binom{\mathrm{w_H}(x')}{2} + \mathrm{w_H}(x')$ and get:

$$(-1)^{e_n}(-1)^{\mathbf{e}'\cdot x' + \mathrm{w_H}(x') + \binom{\mathrm{w_H}(x')}{2}} \mathsf{Z}\Big(|0\rangle + (-1)^{\mathrm{w_H}(x')} |1\rangle\Big) \tag{6.12}$$

We then remark that $\mathrm{w_H}(x') \pmod 2 = \bigoplus_{j\in[n-1]} x_j = \mathbf{1}_{n-1} \cdot x'$ where $\mathbf{1}_{n-1}$ is the vector of length $n-1$ containing 1 in every position. Therefore the expression above is equal to:

$$(-1)^{e_n} \mathsf{Z} (-1)^{\mathbf{e}'\cdot x' + \mathbf{1}_{n-1}\cdot x' + \binom{\mathrm{w_H}(x')}{2}} \Big(|0\rangle + (-1)^{\mathrm{w_H}(x')} |1\rangle\Big) \tag{6.13}$$

We denote $\overline{\mathbf{e}'} = \mathbf{e}' \oplus \mathbf{1}_{n-1}$ the vector where every bit of $\mathbf{e}'$ has been flipped. Since $\mathbf{e}' \cdot x' + \mathbf{1} \cdot x' = (\mathbf{e}' \oplus \mathbf{1}_{n-1}) \cdot x' = \overline{\mathbf{e}'} \cdot x'$,we get that the term for $x_n = 1$ is equal to $\frac{(-1)^{e_n}}{\sqrt{2}} \mathsf{Z} |\phi_{n-1,d,\overline{\mathbf{e}'}}\rangle$. Recalling that the term for $x_n = 0$ was equal to $|\phi_{n-1,d,\mathbf{e}'}\rangle$, we get that:

$$|\phi_{n,d,\mathbf{e}}\rangle = \frac{1}{\sqrt{2}} \big(|\phi_{n-1,d,\mathbf{e}'}\rangle + (-1)^{e_n} \mathsf{Z} |\phi_{n-1,d,\overline{\mathbf{e}'}}\rangle\big) \tag{6.14}$$

Replacing $\mathbf{e}$ with $\bar{\mathbf{e}}$:

$$|\phi_{n,d,\bar{\mathbf{e}}}\rangle = \frac{1}{\sqrt{2}} \big(\, |\phi_{n-1,d,\overline{\mathbf{e}'}}\rangle + (-1)^{e_n+1} \mathsf{Z} |\phi_{n-1,d,\mathbf{e}'}\rangle\big) \tag{6.15}$$

Therefore (where $\mathbf{e}''$ is equal to $\mathbf{e}'$ with the least significant bit e_{n-1} removed):

$$\begin{aligned} |\phi_{n-1,\mathbf{e}'}\rangle &= \frac{1}{\sqrt{2}} \big(|\phi_{n-2,\mathbf{e}''}\rangle + (-1)^{e_{n-1}} \mathsf{Z} |\phi_{n-2,\overline{\mathbf{e}''}}\rangle\big) \\ (-1)^{e_n} \mathsf{Z} \Big|\phi_{n-1,\overline{\mathbf{e}'}}\Big\rangle &= \frac{1}{\sqrt{2}} \big((-1)^{e_n} \mathsf{Z} |\phi_{n-2,\overline{\mathbf{e}''}}\rangle + (-1)^{e_n+e_{n-1}+1} |\phi_{n-2,\mathbf{e}''}\rangle\big) \end{aligned} \tag{6.16}$$

Reinjecting these identities into the one for $|\phi_{n,d,\mathbf{e}}\rangle$ we get:

$$|\phi_{n,d,\mathbf{e}}\rangle = \frac{1}{2} \big((1 - (-1)^{e_n+e_{n-1}}) |\phi_{n-2,d,\mathbf{e}''}\rangle + (-1)^{e_n} (1 + (-1)^{e_n+e_{n-1}}) \mathsf{Z} |\phi_{n-2,d,\overline{\mathbf{e}''}}\rangle\big) \tag{6.17}$$

[3] With the convention that $\{0,1\}^0 = \{\varepsilon\}$ contains only the empty word.

This simplifies to:

$$|\phi_{n,d,\mathbf{e}}\rangle = \begin{cases} \mathsf{Z}\,|\phi_{n-2,d,\overline{\mathbf{e}''}}\rangle & \text{if } e_n \oplus e_{n-1} = 0 \\ |\phi_{n-2,d,\mathbf{e}''}\rangle & \text{if } e_n \oplus e_{n-1} = 1 \end{cases} \tag{6.18}$$

Recall that the result for $n = 0$ was $|\phi_{0,d,\epsilon}\rangle\,\mathsf{Z}^d\,|+\rangle$. It is simple to see that for $n = 1$, $|\phi_{1,e_1}\rangle \in \{|0\rangle, |1\rangle\}$ since entangling two $|+\rangle$ states using CZ and measuring one of them in the $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ basis yields an MBQC computation which is equivalent to applying a Hadamard gate on the input qubit. We can conclude by induction that if n is even then $|\phi_{n,\mathbf{e}}\rangle \in \{|+\rangle, |-\rangle\}$ and if n is odd then $|\phi_{n,\mathbf{e}}\rangle \in \{|0\rangle, |1\rangle\}$ for all values of $\mathbf{e}$.

Knowing this, it is possible to deduce from $|\phi_{n,\mathbf{e}}\rangle = \frac{1}{\sqrt{2}}\big(|\phi_{n-1,\mathbf{e}'}\rangle + (-1)^{e_n}\mathsf{Z}\,|\phi_{n-1,\overline{\mathbf{e}'}}\rangle\big)$ that if n is even we have $|\phi_{n,\mathbf{e}}\rangle = \big|\phi_{n,\overline{\mathbf{e}}}\big\rangle$ and if n is odd then $|\phi_{n,\mathbf{e}}\rangle = \mathsf{X}\big|\phi_{n,\overline{\mathbf{e}}}\big\rangle$.[4] This reduces the previous recurrence relation to (for even and odd values of n respectively):

$$\begin{aligned} |\phi_{2k,d,\mathbf{e}}\rangle &= \mathsf{Z}^{e_n+e_{n-1}+1}\,|\phi_{2k-2,d,\mathbf{e}''}\rangle = \mathsf{Z}^{k+\sum_{j\in[2k]} e_j}\,|\phi_{0,d,\varepsilon}\rangle = \mathsf{Z}^{e+d+k}\,|+\rangle \\ |\phi_{2k+1,d,\mathbf{e}}\rangle &= \mathsf{X}^{e_n+e_{n-1}+1}\,|\phi_{2k-1,d,\mathbf{e}''}\rangle = \mathsf{X}^{k+\sum_{j=2}^{2k+1} e_j}\,|\phi_{1,d,e_1}\rangle = \mathsf{X}^{k+\sum_{j\in[2k+1]} e_j}\,|\phi_{1,d,0}\rangle \end{aligned} \tag{6.19}$$

It is easy to verify that $|\phi_{1,d,0}\rangle = |0\rangle$ regardless of the value of d (via the same MBQC computation on two qubits as above). Therefore:

$$|\phi_{2k+1,d,\mathbf{e}}\rangle = \mathsf{X}^{e+k}\,|0\rangle = \mathsf{H}\mathsf{Z}^{e+k}\,|+\rangle \tag{6.20}$$

These formulae can be regrouped as follows, yielding the desired result:

$$|\phi_{n,\mathbf{e}}\rangle = \mathsf{Z}^d\mathsf{H}^n\mathsf{Z}^{e+\lfloor\frac{n}{2}\rfloor}\,|+\rangle \tag{6.21}$$

■

We can now analyse the knowledge of a coalition of malicious Clients and Server. The Server receives from any honest Client j a BB84 state that depends on two hidden parameters a_j and b_j. These parameters are never revealed and so the state is the perfectly mixed state from the Server's point of view. Since this is the only communication from honest to malicious players, the protocol is automatically perfectly blind with regard to these parameters.

We now look at the dependency of the final state on these hidden values in the case where only a fixed Client j is honest. At the end of the procedure, the parity of n is classically One-Time-Padded by the values a_j and so the basis of the output state is not revealed. Furthermore, if n is even then the final state is $\mathsf{Z}^{e+d+\frac{n}{2}}\,|+\rangle$. Since b_j is hidden, the parity of the parameter $e + d + \frac{n}{2}$ is classically One-Time-Padded either by $d = \sum_{i\in D} b_i$ if the Client has sent a dummy or $e = \sum_{i\in R} b_i + c_i$ otherwise.

[4] Otherwise there are value of e_n for which the state can be null in the recurrence relation.

Similarly if n is odd, the state is given by $\mathsf{X}^{e+d+\lfloor * \rfloor \frac{n}{2}} |0\rangle$ and the same reasoning applies. This means that the output state is completely mixed from the point of view of the malicious parties as long as the values a_j and b_j of the honest Client remain secret.

Equivalently, the honest Client can send half of an EPR-pair to the Server and measure its own half either in the basis $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ or $\{|+\rangle\langle +|, |-\rangle\langle -|\}$ depending on its secret parameter a_j.[5] This measurement can be postponed until the end of the protocol or even later without the Server being able to distinguish this situation from the protocol's execution.

6.3.3 Double Blind Quantum Computation Protocol

We now show how to construct the Double-Blind Quantum Computation Resource used as a black-box in the construction of the DMPQC Protocol in the previous Section. The aim of the DBQC Protocol is to allow a computation, known only to a classical third party called the Orchestrator, to be performed by a Server on quantum inputs supplied by Clients in such a way that neither the Clients nor the Server are able to gain any knowledge about the computation itself and the input states of honest Clients.

High-Level View of the Protocol. Similarly to the construction of the MPQC Ideal Resource, the DBQC Ideal Resource will be assembled from the Double-Blind Rotated State Preparation (Protocol 16) applied to specific states, followed by a regular single-client UBQC Protocol. Protocol 16 is first used to produce a collaboratively encrypted qubit for each non-output vertex in the UBQC computation graph G. The classical outputs are provided to the Orchestrator so that it knows the corresponding key. This is followed by the Server performing entanglement operations given by the edges of G. Before the computation step starts, the Clients privately send to the Orchestrator some random binary values $r_j(v)$ for $v \in O^c$ that will be combined into $r(v) = \bigoplus_j r_j(v)$ to OTP the measurement outcomes obtained by the Server. The Orchestrator chooses the computation to be performed and, given values of $\theta(v)$ and $a(v)$ deduced from the state preparation step, it then instructs the Server to perform single qubit measurements. At the end of the computation step, the Server has a Q-OTPed version of the quantum output of the computation whose encryption key s is given by the values of $\hat{s}^X(v)$ and $\hat{s}^Z(v)$ for each output qubit $v \in O$, which are defined in Section 2.3. The only difference with regular UBQC is that the output is then left encrypted at the Server's interface while the Orchestrator keeps the corresponding decryption key k, which is equal to d.

Protocol 18 presents the formal description of the Double-Blind Quantum Computation for a given unitary U to be implemented on a graph state defined by G and flow f.

We can now state the main result of this section:

Theorem 6.1 (Composability of Double-Blind Quantum Computing)**.** *Under the assumption that at least one Client is honest and that the Orchestrator is honest, Protocol 18 ϵ_D-constructs the Double-Blind Quantum Computing Ideal Resource 20 for $\epsilon_D = 0$ (i.e. perfect security) from Insecure Quantum Channels between each Clients and the Server, Authenticated Classical Channels between all parties and Secure Classical Channels between each Client and the Orchestrator.*

[5] The distribution over its sampled parameter b_j, drawn uniformly at random in the protocol or via this measurement, is the same.

Protocol 18 Double Blind Quantum Computation

Inputs:

- Each Honest Client $j \in [N]$ has a quantum register $\mathcal{X}_j$ which contains their respective part of a collectively possessed state ρ_{inp}.
- The Orchestrator has as input the classical description of a unitary U in the form of a measurement pattern (angles $\{\phi(v)\}_{v\in O^c}$ and flow) on a graph (G, I, O).
- The Server has no input.

Protocol:

1. **State Preparation:** For the input of each Client j, the Clients and Server perform the Double-Blind Rotated State Preparation Protocol 16, with Client j sampling a uniformly random bit $a(v) \in_R \{0,1\}$, applying $\mathsf{X}^{a(v)}$ to its input register $\mathcal{X}_j$ and using it as input. For non-input measured qubits $v \in I^c \cap O^c$ of the graph G, without loss of generality, Client N supplies an input qubit to the Double-Blind Rotated State Preparation Protocol in the state $|+\rangle$. For output qubits, the Server prepares $|+\rangle$ states.
2. **Key Reconstruction:** The Clients send to the Orchestrator the secret parameters used in all executions of the Double-Blind Rotated State Preparation Protocol, along with the values $a(v)$ for their input and a value $r_j(v)$ for all measured qubits of G. The Server sends to the Orchestrator the measurement results of the executions. The Orchestrator deduces from Equation 6.1 the state of all non-input qubits and encryption of inputs for use in the UBQC computation phase and recombines the values hiding the measurement outcome as $r(v') = \bigoplus_j r_j(v')$ for $v' \in O^c$.
3. **Entanglement:** The Server entangles the qubits obtained at the end of the State Preparation according to the edges of the computation graph G with CZ gates.
4. **Computation:** For non-output vertices $v \in O^c$, as in UBQC, the Orchestrator computes the measurement angle $\delta(v) = \phi'(v) + \theta(v) + r(v)\pi$ and sends this angle to the Server. The Server measures location v along the $\{|+_{\delta(v)}\rangle, |-_{\delta(v)}\rangle\}$ basis and returns the measured outcome $b(v)$ to the Orchestrator.
5. **Termination:** The Orchestrator sets as output $k = (\hat{s}^X(v), \hat{s}^Z(v))_{v\in O}$. The Server sets the qubits corresponding to vertices in O as its output. The Clients have no output.

Proof. The goal of the proof is to describe Simulators attached to the interfaces of Malicious parties of the Ideal Resource so that the real-world implementation is indistinguishable from the ideal-world version with the Simulator.

We start by showing that our Protocol is correct when all parties are honest. After the State Preparation and Entanglement steps, the Server holds the same resource state as the one used in the UBQC Protocol presented in Section 3.5.1. Then, the Computation step is identical to the single-client UBQC Protocol with the Orchestrator playing the role of the Client, which implies correctness after the measurements. The Termination step then correctly sets the random Q-OTP key k to the value given by the single-client UBQC Protocol. Therefore, all quantum and classical data produced by the Protocol are identical to the ones returned by the Ideal Resource 20, proving the correctness of Protocol 18.

The second case to examine is when the Server, the Orchestrator and at least one Client are honest. This is a one-way protocol from the malicious Clients to the honest parties, which is always perfectly blind as shown in Theorem 7.2 of [41].

Finally, the case with a malicious Server and possibly some malicious Clients can be dealt with in the worst-case scenario, which corresponds to a single honest Client (eg. Client h) and Orchestrator, while all other parties are malicious and colluding. The corresponding Simulator σ, connected to Malicious Clients and Server on one side, and to the corresponding interfaces of the Ideal Resource on the other

one, is presented in Simulator 10.

We denote with Ω a function from the set of input qubits I, to the set of Clients $[N]$, that determines which Client "owns" the input qubit at location i and is thus supposed to provide it. To mirror the description of the protocol, the domain of Ω is extended to all the vertices in O^c, so that Ω is unchanged on I and is constant and equal to N on $I^c \cap O^c$.[6]

Simulator 10 Malicious Client-Server Coalition

For each $v \in O^c$:

1. The Simulator creates an EPR-pair and sends half of it to the Server.
2. It receives the quantum states from the malicious Clients and forwards them to the Server (i.e. $N-1$ states for each $v \in O^c$).
3. It receives from each malicious Client j, the values for $r_j(v)$ and $\theta_j(v)$ for $v \in O^c$, and for $v \in I$ and $j = \Omega(v)$ it receives $a(v)$.
4. It receives the vector $t(v)$ from the Server.
5. It sends $\delta(v)$ chosen at random in Θ to the Server and receives $b(v) \in \{0,1\}$ in return.
6. It sends to the Ideal DBQC Resource (Resource 20) its remaining half-EPR-pair, along with the values $\theta_j(v)$, $r_j(v)$, $a(v)$, $t(v)$, $\delta(v)$ and $b(v)$.

To show that the Ideal Resource and this Simulator are indistinguishable from the real-world Protocol 18, we need to prove that by observing the classical messages and quantum states exchanged, no Distinguisher can tell apart the two situations. This is done as in [41] by reducing Protocol 18 to one which is equivalent from the point of view of a Distinguisher, and where it will be possible to identify the Ideal Resource and the Simulator. Figure 6.5 depicts the protocol when focusing on a single qubit v owned by an honest Client h.

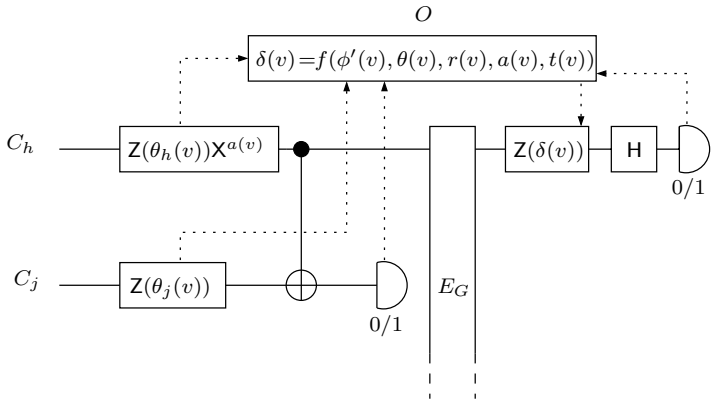


Figure 6.5: Schematic circuit for implementing DBQC with 2 Clients. C_h is honest and owns qubit v whereas C_j provides a $|+\rangle$ state. E_G is the entangling operation involving the other qubits used in the computation. The Orchestrator computes the measurement angle $\delta(v)$ as a function of the target computation and the previous measurements through $\phi'(v)$, of the secret parameters defined by the Clients for qubit v, and of the public measurement outcomes $t(v)$.

[6] The ownership of non-input qubits can be symmetrised across Clients - for example by choosing the owner of these qubits at random - but here the last Client is always chosen as their owner for simplicity of presentation. This can be done without loss of generality since these qubits are prepared the same way regardless of the computation or input: in any case, one client's qubit remains unmeasured for each location and it does not change anything in the protocol or proof to fix it in advance for non-inputs.

We start by replacing the quantum operations performed by Client h in the Double-Blind Rotated State Preparation (Protocol 16). For all non-output qubits, it instead prepares an EPR-pair and sends half of it to the Server. For its non-input qubits it measure the other half in basis $\left|\pm_{-\theta_h(v)}\right\rangle$ for $\theta_h(v) \in_R \Theta$. For its input qubits, it first applies a CNOT between its input (as control) and the other half-EPR-pair (as target) followed by a $\mathsf{Z}(\theta_h(v))$ gate and a Hadamard on the control qubit and then measures both in the computational basis. The measurement results are set as values for $r_h(v)$ (control) and $a(v)$ (target). The rest of the protocol remains the same.

We now analyse its effect on the information accessible to the Distinguisher. Since Client h is honest and never reveals θ, the qubits that it sends to the Server appear completely mixed from the Distinguisher's perspective. Replacing them by half EPR-pairs does not change this. Additionally, the classical information sent in this reduction by Client h to the Orchestrator and later used in determining the measurement angles follow the same probability distribution as in the original protocol while remaining correct. Combining these two facts establishes the equivalence of the original protocol and this first reduction from the Distinguisher's perspective.

The second reduction further alters the modified protocol by (i) delaying in time the measurement on the EPR-pairs for each $v \in O^c$, which does not modify the distribution of quantum nor classical messages exchanged between the various parties; (ii) choosing the angles $\delta(v)$ at random and adapting the angle $\theta_h(v)$ instead of the other way around, which keeps the angle-update equation of UBQC satisfied and $\theta_h(v)$ uniformly distributed over Θ. This *measurement-induced value* of $\theta_h(v)$ is given by:

$$\theta_h(v) = \begin{cases} \text{if } \Omega(v) = h \ : \ \delta(v) - \phi'(v) - \sum\limits_{j \neq h} (-1)^{t_j(v)+a(v)} \theta_j(v) \\ \text{if } \Omega(v) \neq h \ : \ (-1)^{t_h(v)+a(v)} \times \Big(\delta(v) - \phi'(v) - \theta_{\Omega(v)}(v) - \sum\limits_{\substack{j \neq \Omega(v) \\ j \neq h}} (-1)^{t_j(v)+a(v)} \theta_j(v)\Big) \end{cases} \tag{6.22}$$

Here again, the equivalence between the two protocols is immediate: the UBQC equations are still satisfied and no Distinguisher can detect that the measurements have been performed later. The combined modifications are presented in Reduction 1.

Reduction 1 Replacing Q-OTP with Teleportation and Delayed Measurement

Client h:

1. For locations $v \in O^c$, it prepares an EPR-pair and sends half of it to the Server.
2. If $i \in I$ and $h = \Omega(v)$:
   a) It performs a CNOT between its input qubit (control) and the remaining half EPR-pair (target) followed by a Hadamard on the control qubit.
   b) It measures the target qubit in the computational basis and sends the obtained value to the Orchestrator as $a(v)$.

The Orchestrator:

1. It chooses a measurement angle $\delta(v) \in_R \Theta$ and sends it to the Server. It receives in return the measured outcome $b(v)$.
2. It computes $\theta_h(v)$ according to Equation 6.22.
3. It asks Client h to measure the control (yet unmeasured) qubit in the basis defined by $\theta_h(v)$ and receives in return the outcome of the measurement as $r_h(v)$.

The rest of the protocol is unchanged.

It can be checked that the Simulator's operations can be extracted from this modified protocol by grouping the EPR-pair creation done on behalf of Client h with the operations consisting of sending and receiving quantum or classical data to the Server and to the Ideal Resource. To conclude the proof, we need to show that, using the data transmitted by the Simulator, the Ideal Resource is able to take into account the deviation induced by the malicious parties and output the correct encryption key k for the Orchestrator. This is done by isolating an explicit quantum circuit (Ideal Resource 21) from the modified protocol, which then functions as a sub-system of the DBQC Ideal Resource 20. Figure 6.6 depicts the circuit after the reduction and extraction of the Simulator and the Ideal Resource.

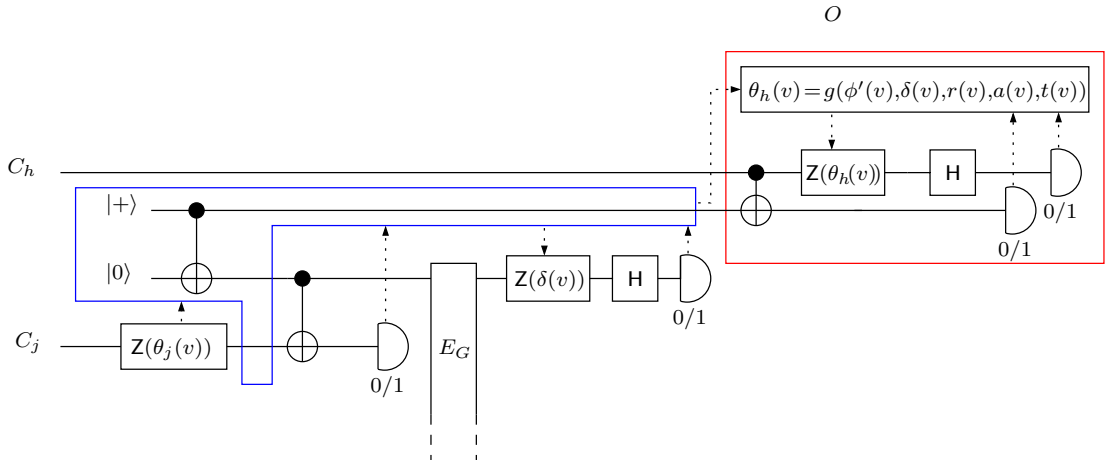


Figure 6.6: Schematic circuit for the reduced protocol where the Simulator (blue) and Ideal Resource (red) have been singled out. From the point of view of C_j and the Server, the interaction with the simulator is the same as for the concrete implementation of the protocol. More importantly, even if a distinguisher provides all input states, comprising that of C_h and O, no difference will be appearent between this setup and the concrete implementation.

Resource 21 Deviation Teleportation

The following steps correspond to the concrete implementation of step 3 of the computation by the DBQC Ideal Resource 20:

1. The Ideal Resource performs a CNOT between the input qubits of Client h (control) and the corresponding half-EPR-pairs (target) provided by the Simulator σ. It measures the target qubit in the computational basis and gets $a(v)$.
2. For qubits $v \in O^c$, it computes the angles $\theta_h(v)$ according to Equation 6.22. It performs the corresponding measurements in the basis $\left|+_{\theta_h(v)}\right\rangle$ in the order defined by the flow, getting the values of $r_h(v)$ and updating the measurement angles for the next qubits.
3. For qubits $v \in O$ it computes the corrections $s = (\hat{s}^X(v), \hat{s}^Z(v))$ and sends the key $k = s$ to the Orchestrator.

Note that there is no need for the Ideal Resource to transmit additional information to the Simulator as its action leads exactly to $\mathsf{QOTP}_k(\mathcal{E}(\rho_{h,[N]\setminus h,U}))$ being prepared in the Server's output register, where the set $[N] \setminus h$ corresponds to the malicious Clients, the subscript U denotes the Orchestrator's computation description, and $\mathcal{E}$ is the deviation chosen by the Server and the malicious Clients.

This concludes the proof as the Simulator together with the Ideal Resource are shown to stem from a protocol equivalent to the real-world implementation, making both situations perfectly indistinguishable. ■

6.4 Using DBQC to Bootstrap Verification

In this section, we show how the DBQC Protocol can be used to prepare VBQC client-encrypted states. Subsection 6.4.1 give one way of implementing the necessary transformations – i.e. applying H gates on dummies while leaving the other qubits untouched – while subsection 6.4.2 gives constraints on the DBQC computation so that, although it is not verified, any deviation occurring at that stage is independent of the chosen VBQC client-encrypted state and thus will be handled by the VBQC protocol itself. It also shows that our proposal satisfies these additional constrains.

6.4.1 VBQC Client-Encrypted State Preparation Using DBQC

6.4.1.1 Subtleties Regarding DBQC Input Qubit Encryptions

We start by emphasizing some fine points regarding how the DBQC Protocol implements the unitaries used to transform the inputs of all Clients into the VBQC client-encrypted state.

The input states to the DBQC sent by the Clients are of two types: either they correspond to their actual inputs to the DMPQC Protocol, in which case they are Q-OTPed with $\mathsf{Z}(\theta)\mathsf{X}^a$, or they are states of the form $|+_\theta\rangle$ for a random θ of their choice. The latter are destined to become the non-input qubits of the VBQC client-encrypted state prepared by the DBQC Protocol: either dummies, traps or computation qubits.

Hence, some qubits need to be turned into dummies, while the others must remain either $|+_\theta\rangle$ states or encrypted inputs. While this is the goal of the application of DBQC, precautions need to be taken regarding the proper implementation of the operations that are performed during this step. We begin by noting that, since the DBQC Protocol requires an Q-OTP encryption of its inputs using operators of the form $\mathsf{Z}(\theta)\mathsf{X}^a$, it is possible to view a $|+_\theta\rangle$ state in two different ways: it is either a state $|+\rangle$ encrypted with $\mathsf{Z}(\theta)\mathsf{X}^a$ or a state $|+_{\theta'}\rangle$ encrypted with $\mathsf{Z}(\theta-(-1)^a\theta')\mathsf{X}^a$, for an arbitrary choice of angle $\theta' \in_R \Theta$ and bit $a \in_R \{0,1\}$. It is thus possible for the Orchestrator to arbitrarily choose to re-encrypt the states for free by picking a different rotation angle θ'.

Specifically, to prepare the dummies, we consider the $|+_\theta\rangle$ states as encrypted $|+\rangle$ states, upon which the DBQC computation must have the effect of applying H to obtain a state in the computational basis. For the computation and trap qubits on the other hand, we remark that using the angles as in the original UBQC Protocol has the effect of striping away the $\mathsf{Z}(\theta)$ encryption and replacing it with a standard Q-OTP. This is useful for the dummies but must be avoided for other qubits as they have to remain rotated to serve as computation and trap qubits. However, the Orchestrator can at will choose θ' and consider the $|+_\theta\rangle$ state as an encrypted $|+_{\theta'}\rangle$ state. Then the result of the operations performed through DBQC on all trap and non-input computation qubits with this encryption must be equivalent to applying an I gate on these states $|+_{\theta'}\rangle$. Otherwise, if the usual definition is applied, the final states would simply be in the $|\pm\rangle$ basis.[7]

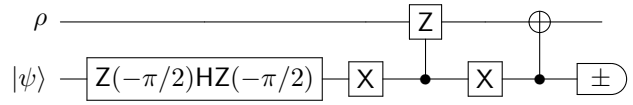
[7] This is required here since the qubits of the last layer are prepared by the Server the state $|+\rangle$.

We remark that using the angles as in the original UBQC Protocol has the effect of striping away the $\mathsf{Z}(\theta)$ encryption and replacing it with a standard Q-OTP. This is useful for the dummies but must be avoided for other qubits as they have to remain rotated. At the beginning of the DBQC Protocol, the DMPQC inputs start off encrypted with operation $\mathsf{Z}(\theta)\mathsf{X}^a$ and they must be similarly encrypted at the beginning of the VBQC execution. We can use the same trick as above for the computation and trap qubits and consider them as encrypted with $\mathsf{Z}(\theta-\theta')\mathsf{X}^{a\oplus a'}\mathsf{Z}\Big((-1)^{a\oplus a'}\theta'\Big)\mathsf{X}^{a'}$, for a random choice of θ' and a'. Then the parameters $\theta-\theta'$ and $a\oplus a'$ are used in the computation phase of DBQC, while $(-1)^{a\oplus a'}\theta'$ and a' are used in the VBQC execution (these need to be further updated first to account for the additional encryption generated through the DBQC Protocol execution).

6.4.1.2 H/I-Gadget

We show below how the Orchestrator can choose to apply the identity or a Hadamard gate blindly to a quantum state using DBQC. We will construct a measurement pattern such that it is the same for each qubit in the DTG (regardless of whether it is an input, computation, trap or dummy) up to the first measurement angle. This is required so that no possible attack in the DBQC depends on secret parameters defining the final VBQC client-encrypted state (see Section 6.4.2 for details).

To this end, we start with the following circuit, where ρ is the arbitrary quantum state that we wish to modify, and $|\psi\rangle$ is used to control which operation is performed:



Above, the first gate $\mathsf{Z}(-\pi/2)\mathsf{HZ}(-\pi/2)$ is a rotated Hadamard gate. It maps the $\{|0\rangle, |1\rangle\}$ eigenvectors of Z onto the $\left\{\left|+_{\pi/2}\right\rangle, \left|-_{\pi/2}\right\rangle\right\}$ eigenvectors of Y and vice-versa. Following the sequence of gates above, for $|\psi\rangle = \alpha\left|+_{\pi/2}\right\rangle + \beta\left|-_{\pi/2}\right\rangle$, the state of the first and second qubits before the measurement is:

$$\alpha\mathsf{Z}(\rho)\,|0\rangle - i\beta\mathsf{X}(\rho)\,|1\rangle \tag{6.23}$$

As a consequence, when $|\psi\rangle$ is $\left|+_{\pi/2}\right\rangle$ it applies Z to the first qubit, irrespectively of the measurement outcome on the second. Similarly, when $|\psi\rangle = \left|-_{\pi/2}\right\rangle$, it applies X to the first qubit.

The same computation can be carried out for $|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}\Big(\left|+_{\pi/2}\right\rangle + i\left|-_{\pi/2}\right\rangle\Big)$. There, a 0-outcome corresponding to the projector onto $\left|+_{\pi/2}\right\rangle$ reduces the state of the first qubit to:

$$\frac{1}{\sqrt{2}}(\mathsf{Z}+\mathsf{X})(\rho) = \mathsf{H}(\rho), \tag{6.24}$$

while the 1-outcome corresponding to the projector onto $\left|-_{\pi/2}\right\rangle$ reduces the state of the first qubit to:

$$\frac{1}{\sqrt{2}}(\mathsf{Z}-\mathsf{X})(\rho) = \mathsf{ZXH}(\rho). \tag{6.25}$$

Finally, when $|\psi\rangle = |-\rangle$, the 0-outcome corresponds to applying ZXH to the first qubit state while the 1-outcome applies H.

Hence, by properly implementing Pauli corrections, the above circuit applies the identity or a H depending on the state of the input for the second qubit. If we further simplify the circuit by commuting

the second X gate for the second qubit through the controlled-NOT gate and absorb the resulting X gate on the first qubit into corrections, and the one on the second qubit into the measurement, we obtain the following corrections

2nd qubit input	Outcome	Correction	Effect
$\|+i\rangle$	0	Y	I
$\|+i\rangle$	1	Y	I
$\|-i\rangle$	0	I	I
$\|-i\rangle$	1	I	I
$\|+\rangle$	0	X	H
$\|+\rangle$	1	Z	H
$\|-\rangle$	0	Z	H
$\|-\rangle$	1	X	H

We can now turn this last version of the circuit into a measurement pattern by recalling that for a graph with 2 connected vertices, when the first qubit in state ρ is measured along an angle θ, the output qubit (if it is initially in the $|+\rangle$ state) is in the state $\mathsf{Z}(\theta)\mathsf{H}(\rho)$. Additionally, we use the identity $\mathsf{XZ}(-\pi/2)\mathsf{HZ}(-\pi/2) = \mathsf{Z}(\pi/2)\mathsf{HZ}(\pi/2)$. We then obtain the measurement pattern from Figure 6.7.

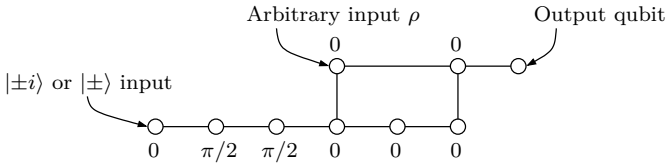


Figure 6.7: MBQC pattern for DBQC computation applying H or I to the arbitrary input qubit in state ρ.

Instead of considering that the second input qubit is in one of the states $\left\{\left|+_{\pi/2}\right\rangle, \left|-_{\pi/2}\right\rangle, |+\rangle, |-\rangle\right\}$, we can fix it to be an encrypted $|+\rangle$ state. Replacing the first measurement angle of the bottom line with angle $-\pi/2$ can then be seen as applying first a $\mathsf{Z}(\pi/2)$ rotation to $|+\rangle$, thus transforming it to $\left|+_{\pi/2}\right\rangle$, before measuring in the $|\pm\rangle$ basis. Hence, by properly choosing the measurement angle of the first qubit of the bottom line, the Orchestrator can choose whether to apply H or I.

To summarise, for all positions in the DTG used in the DMPQC Protocol, the measurement pattern in Figure 6.7 is applied by considering the following possible cases (using the considerations in Section 6.4.1.1):

- If the output qubit in the measurement pattern is a dummy in the DTG, then the upper input qubit is considered to be an encrypted $|+\rangle$ state and the measurement angle of the lower input qubit is chosen by the Orchestrator to be 0.
- If the output qubit is a trap, computation or input, the upper input qubit is treated as either an encrypted $|+_\theta\rangle$ state (trap or non-input computation) or an encrypted input state $\mathsf{Z}(\theta)\mathsf{X}^a(\rho_i)$ and the measurement angle of the lower input qubit is $-\pi/2$.

Finally, anticipating on the result proven in the next Subsection, we remark that the graph used for the computation is invariant with respect to a change of DTG colouring. Additionally, the measurement

angles are in $\{0, \pi/2, \pi, -\pi/2\}$ which implies that, as a Clifford computation, all the measurements during the DBQC Protocol can be performed non-adaptively and sent in a single round of communication. The corrections are taken into account by updating the Q-OTP key encrypting the states after all measurements have been carried out.

6.4.2 Effect of Adversarial Deviation during DBQC on Prepared State

Proving the security of our DMPQC Protocol requires the DBQC Protocol to produce a VBQC client-encrypted state up to a CPTP deviation which is independent of the secret parameters of the state, which consist of the encryption of the state and the colouring of the DTG. Plugging these deviated states in the VBQC Protocol of [78] preserves its original security properties, as its proofs of verifiability and blindness begin precisely with these ""good-enough" states. We show in this Section that applying the DBQC Protocol using the MBQC pattern defined above yields precisely such a state (through Theorem 6.2 and Lemma 6.3). We start by defining good-enough VBQC execution states in Definition 6.1.

Definition 6.1 (Good-Enough VBQC Execution States)**.** *We say that a quantum state is a* good-enough VBQC execution state *if it is of the form $\mathcal{E}(\rho_V \otimes \rho_{aux})$, where $\mathcal{E}$ is a CPTP map depending only on the public and dishonest party parameters, ρ_{aux} is an auxiliary adversarially-chosen state and ρ_V is the state of quantum systems comprising:*

1. *Quantum inputs encrypted by operator $\mathsf{Z}(\theta(v))\mathsf{X}^{a(v)}$, for $\theta(v) \in_R \Theta$ and $a(v) \in_R \{0,1\}$.*
2. *Quantum computation and trap qubits $\left|+_{\theta(v)}\right\rangle$, for $\theta(v) \in_R \Theta$.*
3. *Quantum dummy states of the form $|d(v)\rangle$, for $d(v) \in_R \{0,1\}$.*
4. *Traps, computation qubits (including inputs) and dummies are placed according to a* verifiable graph *G, e.g. the Dotted-Triple Graph [78], which is defined as follows:*
   - *The probability, taken over all possible auxiliary states, for the Server to guess that any qubit in the graph is a trap is lower-bounded by a constant value p_t.*
   - *Trap vertices are linked only to dummy vertices.*
5. *Classical states of the form $|\delta(v)\rangle$ represented as computational basis states on three qubits (as multiples of $\pi/4$).*

In the above, $\delta(v) = (-1)^{a(v)}\phi'(v) + \theta(v) + r(v)\pi + \bigoplus_{\tilde{v}\in N_G(v)\cap D} d(\tilde{v})\pi$, for $r(v) \in_R \{0,1\}$, $\phi'(v)$ the flow-corrected angle on qubit v for a branch of computation corresponding to the Server's measurement outcomes b (in Θ for computation qubits and 0 for trap qubits) and D the set of dummy qubits in the verifiable graph G and $N_G(v)$ the neighbours of qubit v in G.

This definition captures the idea of a state that, at the end of an execution of the VBQC Protocol (i.e. for a branch of the computation fixed by the Server's measurements), allows an honest party to verify the correct application of this VBQC computation by checking the traps using secret parameters $a(v)$, $\theta(v)$ and $r(v)$. While this definition is given here for the specific framework used in this work, it could potentially be generalised to other verifiable protocols.

We prove that the state prepared by the DBQC Protocol 18 is indeed a good-enough VBQC execution state. More generally, we give a framework for DBQC computations that lead to good-enough VBQC

execution states in Theorem 6.2 using Clifford computations. On the other hand we conjecture that this cannot be extended to arbitrary quantum computations, because it would imply we can reduce any verification protocol to a protocol where only the last layer is trappified (by doing the actual computation in the preparation phase).

This result formalises the intuitive notion that the honest Client has provided enough randomness to extract a good-enough VBQC execution state from the state prepared by the DBQC Protocol, despite potential meddling from malicious parties.

Theorem 6.2 (Constraints for Good-Enough State Preparation through DBQC)**.** *The output of DBQC Protocol 18 using an MBQC pattern defined by a fixed graph $G = (V, E, I, O)$ with flow $(f, \preceq)$ and base angles $\{\phi(v)\}_{v\in O^c}$ is a good-enough VBQC execution state for the Clients' desired computation if:*

- *For an honest input to the DBQC Protocol, the honest application of the measurement pattern produces a correct VBQC client-encrypted state ρ_{DTG} along with measurements angles $\delta(v)$ that correspond to the correct desired computation for any fixed computation branch.*
- *All measurement angles for the DBQC computation are Clifford, i.e. $\phi(v) \in \{k\pi/2\}_{0\leq k\leq 3}$.*
- *The graph $G = (V, E, I, O)$, the flow $(f, \preceq)$ and all angles associated to vertices that have an X-dependency according to the chosen flow are all invariant under a change of colouring of the DTG.*

Proof. We decompose the proof in two parts: first the Double-Blind Rotated State Preparation Protocol 16, followed by the remaining computation steps of the DBQC Protocol. We unitarise the protocol in order to commute the Adversary's attack and show that the resulting equivalent attack does not depend on the secret parameters. We prove that, irrespective of the adversarial scenarios, the DBQC Protocol produces good-enough VBQC execution states spanning the full range of parameters.

Preparation Phase of DBQC. We consider each prepared qubit v separately and two Clients. The goal of this step is to decorrelate the malicious and honest parameters and their corresponding states. This is achieved by unitarising the state preparation step of the DBQC Protocol and the deviation of the Adversary, which then allows us to "extract" various unitaries from the Adversary's deviation for each prepared qubit.

Crucially, these unitaries extracted from the attack depend only on public or leaked parameters and therefore do not create a dependency of the attack on the secrets of the honest party. They will however depend on the malicious or honest nature of the owner of the prepared qubit. This is allowed since the deviation of the Adversary may anyway depend on this factor. Note that the conditions above also mean that this extraction can be undone by the Adversary and therefore it does not change the set of possible deviations (up to a relabelling). There are three cases to consider and the same unitary can be extracted for all qubits of the protocol in each sub-case, meaning that the final attack does not depend on the positioning of the different types of qubits (computation, trap or dummy). Generalisation for more than two clients can be done by extracting the same operators for each client consecutively.

First, if the owner of the qubit is malicious and the prepared qubit is its input to the VBQC execution, the Server receives the state $\tilde{\rho}$ from the dishonest owner $\Omega(v)$ of vertex v (encrypted with its own

parameters), and a state $\left|+_{\theta_j(v)}\right\rangle$ from honest Client $j \neq \Omega(v)$. The Server also receives classical states $\delta(v)$ and $t_j(v)$, where:

$$\delta(v) = (-1)^{a(v)}\phi(v) + \theta_{\Omega(v)}(v) + (-1)^{t_j(v)}\theta_j(v) + r_{\Omega(v)}(v)\pi + r_j(v)\pi \tag{6.26}$$

In this case, any deviation at this position can be translated to a modification of the malicious player's input, which is always allowed. However the commutation required to incorporate any effect of the deviation into the input also depends on the value of $(-1)^{t_j(v)}\theta_j(v)$ from the honest player since it is used to encrypt this input. This value is already public for these qubits since the values of $\phi(v)$, $a(v)$, $\theta_{\Omega(v)}(v)$ and $r_{\Omega(v)}(v)$ are known to the malicious Client, therefore $(-1)^{t_j(v)}\theta_j(v)$ is known up to a π rotation and the Server can measure the honest Client's qubit to recover $r_j(v)$.[8]

The next case deals with a dishonest owner of vertex v – that is not its input qubit to the VBQC execution – sending the state $|\chi\rangle$ and an honest Client $j \neq \Omega(v)$ providing $\left|+_{\theta_j(v)}\right\rangle$ to the Server. The Server also receives classical states $\delta(v)$ and $t_j(v)$, where:

$$\delta(v) = \phi'(v) + \theta_{\Omega(v)}(v) + (-1)^{t_j(v)}\theta_j(v) + r_{\Omega(v)}(v)\pi + r_j(v)\pi \tag{6.27}$$

The DBQC Protocol instructs the Server to perform a CNOT between the qubits of the owner and the honest party. Having unitarised the protocol and the attack of the Adversary, we can extract the unitary in the dashed box in Figure 6.8 from the general attack operator that follows.

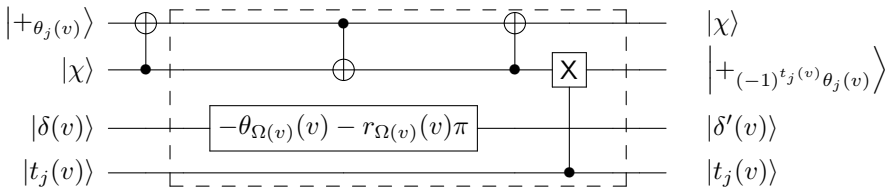


Figure 6.8: The honest Client controls the first line and the owner the second one. The operation acting on the state $|\delta(v)\rangle$ is the unitary associated with the classical (reversible) operation that adds the value $-\theta_{\Omega(v)}(v) - r_{\Omega(v)}(v)\pi$ to any angle in Θ. We have $\delta'(v) = \phi'(v) + (-1)^{t_j(v)}\theta_j(v) + r_j(v)\pi$. The operations inside the dashed boxed are extracted from the attack operation.

In the last case, the owner of vertex v is honest, sending a qubit in state ρ with $\rho = |+\rangle$ for non-inputs, and the other Client is dishonest. Again we have (with $a(v) = 0$ for non-inputs):

$$\delta(v) = (-1)^{a(v)}\phi'(v) + \theta_{\Omega(v)}(v) + (-1)^{t_j(v)}\theta_j(v) + r_{\Omega(v)}(v)\pi + r_j(v)\pi \tag{6.28}$$

As previously, we extract a unitary from the attack (see Figure 6.9).

The same considerations as above apply. In all cases, the final state has the form described in properties 1, 2, 4 and 5 of a good-enough VBQC execution state (Definition 6.1). The proof generalises

[8] Although the malicious Client also know the value of $\phi(v)$ for all computation and trap qubits of the Dotted-Triple Graph, the same reasoning does not apply since it does not know where the nature of the qubits in base-locations that are not associated to its input.

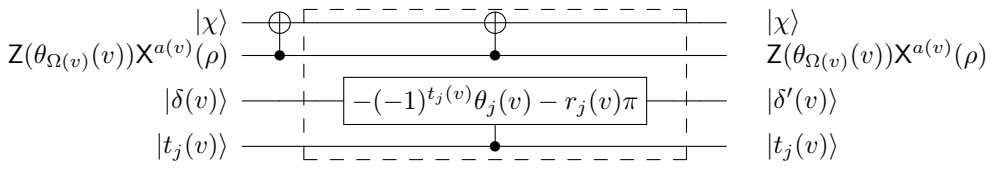


Figure 6.9: The malicious Client controls the first line and the honest owner the second one. The operation on $|\delta(v)\rangle$ is defined as above, but it is now controlled by the value for $t_j(v)$. We have $\delta'(v) = (-1)^{a(v)}\phi'(v) + \theta_{\Omega(v)}(v) + r_{\Omega(v)}(v)\pi$.

to more than two parties by noticing that we can always extract an attack that disentangles the qubit of the owner from the qubits of the other dishonest parties and also undo their corresponding θ's and r's from δ.

The result at the end of this step is a set of qubits in the Server's register that contain: (i) the inputs of all parties to the VBQC computation; (ii) all non-input qubits that will be part of the VBQC graph, some of which must be turned into dummies; (iii) all additional qubits used in the DBQC computation; (iv) for a given branch of VBQC and DBQC computations, the classical state corresponding to the measurement angles sent as instructions to the Server. All states apart from the VBQC inputs of the malicious Clients are encrypted solely with the honest player's parameters, even in the case of a single non-deviating Client.

We can furthermore extract from the Server's deviation the unitary entangling operation for the DBQC graph, after which we can consider the operations applied during the DBQC computation phase, i.e. blindly transforming the states $|+\rangle$ into dummies.

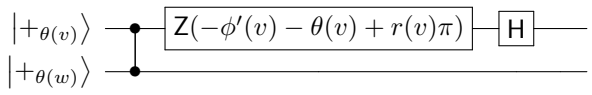
Residual Deviation on a Single Gate Teleportation. We start by determining the residual deviation when a single gate teleportation step is extracted from a general attack.

The corresponding circuit for a single gate teleportation is represented graphically on Figure 6.10a, where the top qubit is the input and the bottom the output. We apply the entanglement operation followed by $\mathsf{Z}(-\phi'(v) - \theta(v) + r(v)\pi)$ and H to the first qubit. This yields a unitarised version of the DBQC computation step, where it is understood that any attack that could have been performed during the preparation phase or during this step has been commuted to after the execution of the circuit and has been reduced as a pure deviation. By simplifying and rearranging these operations we get Figure 6.10b. Because the state after the CZ is stabilized by $(\mathsf{X} \otimes \mathsf{Z})^{p(v)}$ for any choice of $p(v) \in \{0, 1\}$, we get Figure 6.10c after commuting the stabilizer through the Z rotation and the Hadamard. Due to the commutation, the rotation angle is modified and transformed to $-(-1)^{p(v)}\phi'(v)$. We can then get rid of the gate $\mathsf{Z}^{p(v)}$ on the second qubit because it only contributes towards a global phase (Figure 6.10d).

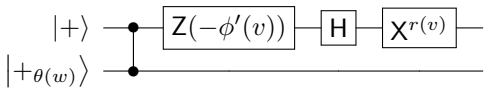
Following [71], by taking the sum over $r(v)$ and $p(v)$, we can use the Pauli Twirling Lemma 2.5 from [34]. This allows to rewrite the residual deviation as a convex combination of Pauli operators on the first qubit tensored with possibly different generic attacks on the output qubit. Because the first qubit is measured in the computational basis, the deviation's effect reduces to a convex combination of no action on the measurement result $b(v)$ tensored with a generic attack on the output qubit and a classical bit flip on $b(v)$ tensored with a possibly different generic attack on the output qubit.

Note that in the original verifiability proof [53], there is no need for these extra parameters $p(v)$. It

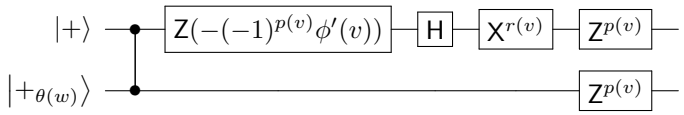
is only required here because we want to assess the effect of a deviation on internal qubits of DBQC instead of computing the probability of obtaining a correct measurement outcome for traps. This can concludes that the most general attack on these qubits are bit flips of the measurement outcomes.



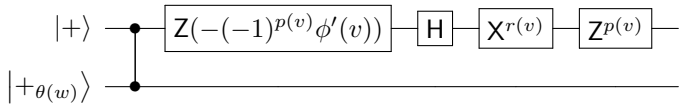
(a) DBQC Computation Step.



(b) Simplification.



(c) Stabilisation by $\mathsf{X}^{p(v)} \otimes \mathsf{Z}^{p(v)}$.



(d) Final Circuit.

Figure 6.10: Elementary step in DBQC computation phase.

Residual Deviation on the Whole DBQC Computation Phase. The above transformation can be applied successively to the whole DBQC computation phase in order to describe the residual deviation for a generic attack. In doing so, the deviation is reduced to a probabilistic combination of bit flips on the classical measurement outcomes $b(v)$, each tensored with (possibly different) arbitrary attacks on the output qubits. In addition, it was shown in the previous section that the residual deviation from the state preparation step was dependent only on public or leaked parameters. Because of the blindness of the scheme, the attacks that can be performed for each gate teleportation are also only depending only on public or leaked parameters. As a consequence, when the residual deviation is written as a convex combination of bit flips for the classical measurement outcomes $b(v)$, each remaining generic attack on the output qubits depends only on public or leaked parameters.

Hence, if there is a dependency upon secret parameters, it can only come from the effect of the classical bit flips.

Effect of the Classical Bit Flips. Finally, we have to consider how the classical bit flips for the measurement outcomes will translate into an attack on the output qubits only. Their effects can be

analysed using the flow of computation. This can be done easily by recalling how the measurement angle of a not-yet-measured qubit w depends on a previous outcome $b(v)$. We have:

$$\begin{aligned} \delta(w) &= (-1)^{s^X(w)}\phi(w) + s^Z(w)\pi + \theta(w) + r(w)\pi, \text{ with} \\ s^X(w) &= \bigoplus_{v \in D^X(w)} b(v) \oplus r(v) \\ s^Z(j) &= \bigoplus_{v \in D^Z(w)} b(v) \oplus r(v) \end{aligned} \tag{6.29}$$

This means that whenever $v \in D^Z(w)$, the bit flip triggers a Z rotation on qubit w, while whenever $v \in D^X(w)$, a bit flip has the effect of flipping the sign of $\phi(w)$. Since $\phi(w) \in \{0, \pi/2, \pi, -\pi/2\}$, this is equivalent to a Pauli attack operator on the output of the DBQC computation phase (updating the Pauli frame).

Independence of secret parameters. Without further constraints, the above residual deviation could very well depend on secret parameters. To understand why, we take a step back and examine the generic procedure to prepare good-enough VBQC execution states using DBQC. It amounts first to collectively encoding quantum inputs and $|+\rangle$ states. Then, after choosing a colouring for the Dotted-Triple-Graph, the Orchestrator chooses a graph and a measurement pattern on it that will implement a unitary for turning some $|+\rangle$ states into dummies and placing all the qubits at their required location.

Because the sets D^Z and D^X above are specific to the chosen DBQC computation graph and its flow, the attack can only be independent of the secret parameters under the condition that the graph used in DBQC for all Dotted-Triple Graph qubits is the same and that the same flow of computation is chosen on this graph (i.e. they are both invariant under permutation of the qubits of the Dotted-Triple Graph in each base-location).[9] By simply imposing this, we already get that the Z correction is independent of any secret parameter since it will always add a π rotation to the measurement angle and so this will affect similarly any qubit regardless of its type.

The X dependency induces additional restrictions. When $v \in D^X(w)$ the effect of the bit flip on $b(v)$ on the subsequently measured qubit w depends on the value of the angle $\phi(v)$. As a consequence, its action might pick up a dependency on a secret parameter of the DBQC computation. We therefore need to analyse it further since this remaining attack is a Pauli attack on the output qubits that cannot be further reduced by twirling. For $\phi(w) \in \{0, \pi\}$ a sign flip has no effect, while there is an additional Z operation when $\phi(w) \in \{-\pi/2, \pi/2\}$. Hence, to get the independence upon the secret parameters (here the value of $\phi(w)$), it is necessary for the angles $\phi(w)$ for w such that $D^X(w) \neq \emptyset$ to be independent on the chosen graph colouring, meaning that the measurement angles of those qubits is invariant under permutation of the qubits in each base-location of the Dotted-Triple Graph.[10]

Only qubits that are in the future of other qubits (in terms of flow) may get a malicious correction. This is directly reflected in the invariance condition since only qubits that are in the image set of the

[9] Intuitively, if the graph or the measurement order induced by the flow is different, the Adversary can already distinguish the qubits in each base-location, but the flow condition is new since it has been proven the an Adversary cannot distinguish between the application of two different flows [97].

[10] This condition is sufficient but not necessary. Instead we can use the fact that the initial positions of the types of qubits (input/computation and traps) is chosen at random in each base-location. Then a less stringent condition on the DBQC computation angles would be to impose that they are either equal or random for each base-location. Summing over this random choice would give an attack CPTP operator that is independent of the input/computational and trap qubits.

flow function can be affected. Therefore, for w such that $D^X(w) = \emptyset$, there is no constraint on the measurement angle as the impact of the bit flips only affects dependent qubits.[11]

This concludes our proof as it shows that the residual deviation on the output qubits – i.e. the prepared good-enough VBQC execution states – is independent of the secret parameters defining the underlying VBQC state.

■

Theorem 6.2 above only guarantees that the attack depends on public and adversarial parameters. The public parameters consist of the graph and flow but may or may not include additional information about the input to this computation or measurement angles. We now show that the computation described in Section 6.4.1.2 both satisfies the constraints of this theorem and also does not leak additional information about the honest player's input which would allow the Adversary to attack it.

Lemma 6.3 (Secure Multi-Client VBQC State Preparation)**.** *Assume that the base graph chosen to perform an MBQC computation through VBQC has degree* 1 *on all input vertices. Apply the DBQC Protocol using the MBQC pattern from Figure 6.7 for each qubit of the final DTG as described in Section 6.4.1.2 on an input state constructed as: (i) MBQC input states and* $|+\rangle$ *states for non-input positions in the DTG and (ii) the position of the MBQC input of honest players is randomly chosen in its associated input base-location. The final state then consists of* $\mathcal{E}(\rho_V \otimes \rho_{aux})$*, where* ρ_V *is a correctly-prepared VBQC client-encrypted state,* ρ_{aux} *is an arbitrary auxiliary state prepared by the Adversary and the deviation* $\mathcal{E}$ *is independent of* ρ_V*.*

Proof. We first need to show that the operation applied through the DBQC Protocol satisfies the conditions given by Theorem 6.2. The correctness of this scheme follows directly from the discussion found in Section 6.4.1.2: it correctly produces computation qubits, traps and dummies in positions defined by a Dotted-Triple Graph colouring, the angles of the graph are all Clifford, the full graph is obtained by applying the same graph and flow on each qubit of the DTG. Therefore it does not depend on their nature or secret parameters. Finally the measurement angles are all the same for each of these applications apart from the lower input, which can be either 0 or $-\pi/2$ depending on the nature of the final qubit. However, this qubit is not X-dependent on any other. Hence, we can apply Theorem 6.2. After this step, we thus have a state of the form $\mathcal{E}(\rho_V \otimes \rho_{aux})$, with a deviation $\mathcal{E}$ that acts on a correctly-prepared VBQC client-encrypted state ρ_V and which is guaranteed to not have picked-up any additional dependency on the secret parameters as a consequence of the application of the DBQC Protocol.

As a second step, we therefore need to assess the public or leaked information available to an Adversary in the rest of the Protocol. We treat the sub-cases corresponding to non-input and input base-locations separately.

For non-input base-locations, it is public knowledge that all qubits used as inputs to the DBQC Protocol are rotated $|+\rangle$ states. The rotation angles of all qubits in the state remain secret as a result of the blindness of the Protocol thanks to the collaborative encryption and the state provided by the honest

[11] Most commonly, all qubits apart from the input qubits have an X dependency, in which case the only angles that may differ according to the nature of the qubits in each base-location are the ones associated to the input positions of the DBQC graph

Client.[12] For the same reason, this is also the case for the measurement angle of the first lower input qubit in the H/I-Gadget above. Hence, an attack cannot depend on the secrets of the honest clients, nor on type – i.e. trap, computation, dummy – chosen by the Orchestrator for each qubit in a non-input base-location.

For input base-locations, if the owner is an honest Client who chose the location of its inputs at random, and because DBQC hides the type chosen by the Orchestrator for each non-input qubit in this input base-location, the Adversary does not gain any information compared to a plain VBQC execution. The Adversary's only additional information comes from its knowledge of the position of inputs provided by the malicious Clients. We now prove that any attack after the state has been prepared that uses this information can be expressed as a modification of the malicious party's input.[13] To this end, we use the fact that all attacks in VBQC can be reduced to classical attacks (flipping the measurement outcomes) by using the twirling lemma[14] and therefore analyse the effect of these only.

These classical attacks are probabilistic mixtures of bit strings that depend on the knowledge of the Adversary. Although the malicious owner of the input qubit knows where this input qubit is placed in its input base-location, it does not know the position of the computation qubits in the second layer of primary base-locations and after that. This means that the state from the the second layer onward is exactly the same from the point of view of the Adversary as one in a normal VBQC execution. Therefore, any attack that does not affect the first layer (malicious) qubits and the added qubits in the connecting edge is equivalent to one in the normal VBQC execution where the Adversary simply decide begin attacking after these positions have already been measured. Hence, the only attacks stemming from this information that are not already taken into account in the analysis of the VBQC Protocol are ones which have a non-trivial effect on this input base location and/or the edge base-location directly connected to it. We now analyse the effect that flipping a measurement outcome on these qubits has on the rest of the state.

A classical attack on the malicious Client's input base location that does not trigger a trap is, by construction, equivalent to a modification of this player's input (by applying Z before performing the entangling and measurement). On the added qubits, the Adversary knows that the neighbours of the computational input qubit cannot be traps since the computation primary qubits are linked to one added computational qubit and two dummies. It can then flip the measurement outcomes of these qubits while being sure to never triggering the trap in this base-location. Flipping the measurement outcome of the dummies has no effect since it is perfectly random and does not influence the rest of the graph through corrections. However, any classical attack on the computational (bridge) qubit translates to a Pauli Z operations on the computational neighbours due to the bridge operation corrections. The Z operation on the input computation qubit can be directly rewritten as a modification of this input since it commutes with the CZ entangling operation. The other Z correction (on the computation qubit of the second layer) is equivalent to having applied an X operation on the input qubit once before entangling the state and then another time after applying the CZ. The first X operation can be integrated as an input modification and the second one is similar to multiplying the measurement angle ϕ associated to this input position by -1. Since this angle is known to the malicious Clients (they know the computation

[12] In a sense, it is precisely because this honest state contains sufficient randomness that we are able to rewrite the final DBQC state in the form of ρ_V up to the subsequent deviation.

[13] This behaviour is always allowed as all Parties are free to choose their input arbitrarily.

[14] Note that the twirling lemma only requires blindness, which is provided by the collaborative encryption.

being performed), this is then equivalent to having pre-rotated the input around the Z-axis by -2ϕ first and applying the correct measurement.

Combining these steps shows that, the knowledge of the location of the malicious Client's inputs can be incorporated into modifications of their inputs, hence showing that the final state $\mathcal{E}(\rho_V \otimes \rho_{aux})$ is effectively a correctly prepared VBQC client-encrypted state up to an arbitrary deviation which is independent of the secret parameters of the state and in particular of the corresponding colouring of the DTG.

■

Using Lemma 6.3 allows us to reuse the security analyses of [78] for our composed protocol, therefore proving the full security of the DMPQC Protocol 19. Furthermore, the general formulation of Theorem 6.2 gives conditions to satisfy for designing other, possibly more efficient, good-enough VBQC execution states preparation protocols.

Colouring-Independence Condition of DBQC Angles is Necessary. We describe here how it is possible to perform an attack on VBQC if the last condition of Theorem 6.2 regarding the measurement angles is not satisfied (i.e. if some measurement angles with a non-trivial dependency set are not independent of the DTG colouring). For this purpose we give a description of a Hadamard/Identity gadget that does not fulfil this condition and show how to use it to break the verifiability of the VBQC Protocol.

The Hadamard or Identity can be applied by using a line graph of five qubits, with the first one being the input and the last on in the line being the output (which is the only one left unmeasured). Than in order to apply the Identity, the chosen measurement angles on the first four qubits are all equal to 0, while for Hadamard the first one is 0 and the other three are $\pi/2$. If the adversary flip the measurement outcome of the next to last qubit, in both cases the final qubit is affected by an additional Z operation (which the Adversary undoes since it may hit a trap). But only in the case of Hadamard being applied, the final output qubit also gains an X correction. Since this operation is used to prepare dummies, this means that the final value of the dummy is flipped compared to its intended value. In short it is possible for an Adversary to apply an attack that always flips the value of a dummy qubit while leaving computation and trap qubits unaffected.

It may seem at first that this attack does not harms the protocol since it does not affect either traps or computations. However the effect of the attack on the dummies is propagated to adjacent qubits in the Dotted-Triple Graph. Through the application of the CZ gate, an application of X on the dummy is equivalent to applying Z on all qubits which are linked to this dummy in the graph. We now show how to use this fact to perform an attack which may hit computation qubits but is never detected by traps.

Consider a line graph of two qubits and its transformation in a Dotted-Triple Graph. This graph contains two primary locations with three qubits and one added location with nine qubits. It is useless to use the attack above on a primary vertex since the primary dummy is always connected to one trap but no computation qubits. We notice that if a even number of dummies that are linked to the same qubit are attacked, then this qubit is not affected by the attack since the effect of the Z operations cancel out. However if an odd number of dummies linked to a qubit are attacked, then this qubit will be corrupted. The trick then relies on the difference in the number of dummies in the neighbourhood of traps and computation qubits. Traps are only linked to dummies while a computation qubit will always

have at least one other computation qubit among its neighbours. It is possible, as show in Figure 6.11 below, to choose added qubits to attack so that each primary vertex is linked to exactly two attacked middle qubits.

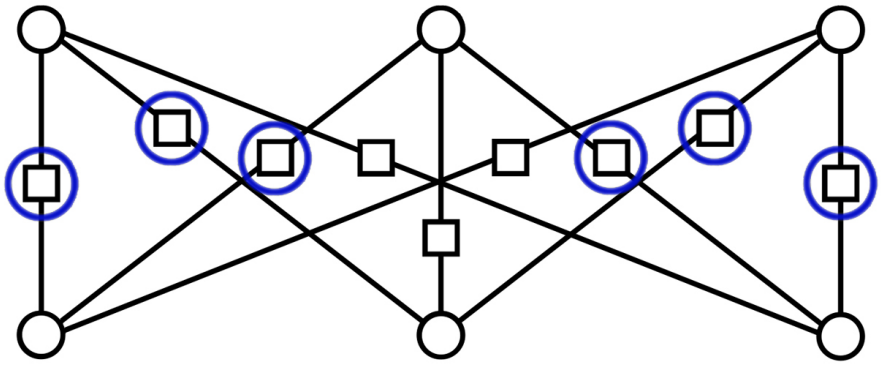


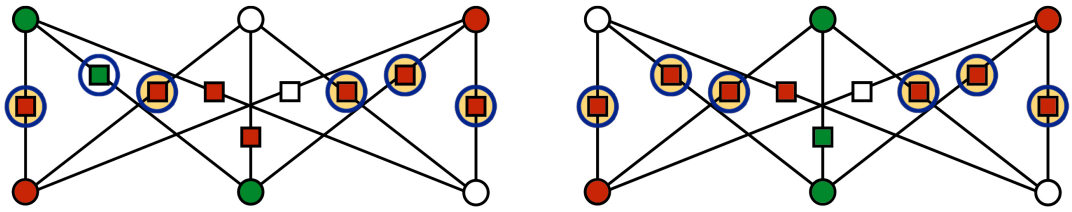
Figure 6.11: Example of attack layout where each top and bottom primary qubit is attached to exactly two attacked added qubits. Qubits that have been chosen for the attack are circled in blue.

In that case, since the primary trap qubits are only linked to dummies, the attack does not trigger either trap (if one of the middle qubits that is attacked is a trap, the effect of the attack on this trap is the Identity as explained above). However, the attack may either affect two dummies linked to the primary computation qubits, in which case there is no attack since the effects cancel out, or one added dummy and the added computation qubit. Then, the effect on the added computation qubit is Identity but the attacked dummies will apply a Z operation on primary computation qubits on both sides of the link. If we assume fixed (but unknown) the attack positions, whether this attack succeeds in modifying the computation depends only on the colouring that is used, while never triggering any trap. The probability of success is equal to 2/3: the attack succeeds if the computational added qubit is chosen for the attack, there are 6 possible choices of attack configuration and each added qubit is left untouched by 2 out of the 6 attack configurations. We give in Figure 6.12 two possible colourings, ones in which the attack has no effect one the computation while the other corrupts it.

Essentially, allowing an attack to depend on the nature of the qubits, even without the Adversary knowing the position of these qubits, introduces new attacks compared to those that are possible in the plain VBQC Protocol. We have shown above that it is sufficient to break the verifiability property of this composed protocol. We have shown in the proof of Theorem 6.2 above that using either fixed or perfectly random angles is sufficient to make this attack independent of the nature of the qubits, but it is unclear what its effect is if the randomisation is not perfect but skewed towards one angle or the other.

6.4.3 Compatibility of Good-Enough States and Proofs of Verifiability

The state after the DBQC Protocol is proven to be equivalent to an honestly prepared VBQC client-encrypted state upon which the Server performs a deviation independent from the secret parameters (Theorem 6.2). This state can therefore be directly used in the VBQC Protocol and all the security



(a) Z attack on both primary computational qubits due to odd number of attacked added dummies.

(b) No attack on either primary computational qubit due to even number of attacked added dummies.

Figure 6.12: Two colourings of the previous graph (computational qubits are green, traps are white and dummies are red) for the same attacked qubits but a different effect on primary computational qubits. Attacked qubits are circled in blue, which translates to an X effect on dummies (yellow-filled circle) and no effect on added computational qubits (empty circle). The primary trap qubits are never affected by the attack since they are always attached to an even number of attacked added dummies.

properties follow since this deviation could have been performed by the Server during the VBQC Protocol as well (and such deviations are taken care of by the verifiability of the protocol).

For the sake of completeness, we show here formally that this good-enough VBQC execution state is sufficient for the proof of verifiability of the VBQC Protocol to go through. We reuse the same notations as in [78], recalled here for clarity, with the subscript denoting dependence on parameters and parenthesis denoting qubit index, e.g. $|\eta_{\nu_T}(v)\rangle$ is a state of qubit with index v that depends on parameters ν_T:

- ν are all random parameters, including the computational random parameters ν_C and the trap random parameters ν_T. The hidden parameters include: all θ's, r's, a's, d's of the honest Client and the positions of the traps T.
- c_r and $C_{\nu_C,b}$ are the classical and quantum corrections on the outputs of the Server.
- $|\Psi_{\nu,b}\rangle$ is the state received by the Server from the Clients and the Orchestrator, including all classical states and quantum states from each party. In particular it contains: The Q-OTPed input qubits with parameters θ and a, all the $|+_\theta\rangle$ states generated from each Client (including the owner for the non-input vertices) for each vertex. $\left|\Psi^{\nu,b}\right\rangle$ also contains all δ's sent by the Classical SMPC for all phases of the protocol corresponding to the fixed branch b. Notice that the δ's have a dependence on T.
- $\mathcal{P}$ is the whole protocol unitary operator applying on $|\Psi_{\nu,b}\rangle$ (the DBQC Protocol 18 and the actual VBQC computation on the latter state).
- Ω is any unitary that represents the deviation on the Server's system, including its private register containing initially the state $\otimes^S |0\rangle\langle 0|$. All deviations at any time step of the protocol have been gathered together. As proven in Section 6.4 for DBQC and since the VBQC Protocol unitary does not depend on ν, there is no dependence of Ω on ν either.

The output state of the protocol, held by the Clients, is the same as Equation (C1) from [78] (with

our choice of $\mathcal{P}$):

$$\text{(6.30)} \qquad \mathrm{Tr}_S \left(\sum_b |b + c_r\rangle\langle b| \, C_{\nu_C,b} \Omega \mathcal{P} (\otimes^S |0\rangle\langle 0| \otimes |\Psi_{\nu,b}\rangle\langle\Psi_{\nu,b}|) \mathcal{P}^\dagger \Omega^\dagger C^\dagger_{\nu_C,b} \, |b\rangle\langle b + c_r| \right)$$

An important point to be made here is that the projection operators $\langle b|$ for the measurements of the Server also include the measurements for the Double-Blind Rotated State Preparation (Protocol 16). Therefore all the t's sent during this step are included in the global b's here. The t's are not encrypted, which means that $c_r = 0$ for these bits. Following the same steps as in [78], we arrive at their Equation (C7):

$$\text{(6.31)} \quad p_{\text{fail}} \leq \sum_{k,b',\nu} p(\nu) \, \mathrm{Tr} \left(\Big(\bigotimes_{v \in T} |\eta_{\nu_T}(v)\rangle\langle\eta_{\nu_T}(v)| \otimes |b'\rangle\langle b'| \Big) \Big(\sum_{i \in E} a_{k,i} \sigma_i \Big) \mathcal{P} \, |\Psi_{\nu,b'}\rangle\langle\Psi_{\nu,b'}| \, \mathcal{P}^\dagger \Big(\sum_{i \in E} a^*_{k,i} \sigma_i \Big) \right)$$

In the equation above: (i) $|\eta_{\nu_T}(v)\rangle = |r(v)\rangle$ for $v \in O^c$ and $|\eta_{\nu_T}(v)\rangle = |\theta(v)\rangle$ for $v \in O$ (the traps in non-output base-locations have already been measured, while the ones in output positions are measured by Clients); (ii) string b' is the substring of b that does not include the measured traps; (iii) E is the subset of all multi-qubit Pauli operators σ_i that can corrupt the output of the computation (the attack unitary Ω becomes a CPTP maps after tracing out the Server's register, expressed in terms of Kraus operators indexed by k, which are then each decomposed as linear combination of Pauli operators σ_i with coefficients $a_{k,i}$).

Before following the proof steps in [78] we need to reduce our state and our protocol to the state and the protocol in [78]. Here we use the fact that, as state before, the DBQC Protocol prepares a good-enough VBQC execution state. We can therefore apply the part of unitary protocol $\mathcal{P}$ that corresponds to the DBQC Protocol and subsequent entanglement of the graph by the Server. This reduces the state before the attack to the same (Dotted-Triple Graph) state as in the VBQC Protocol. Any attack on the ancillary qubits used for this preparation is automatically transferred to the final qubits by a simple re-indexing of the attack operators. Also, an attack on the bridge qubits of the first layer is Pauli twirled using the random pre-rotations of the bridge qubit and remapped as an attack on the input qubit. We then sum over the secret parameters (i.e. unknown to the adversarial coalition) for the non-trap part of the state, including the non-trap δ's. This gives us Equation (C8) from [78] and the rest of their proof follows:

(6.32)

$$p_{\text{fail}} \leq \sum_{k,\nu_T,i,j} a_{k,i} a^*_{k,j} p(\nu_T) \, \mathrm{Tr} \left(\Big(\bigotimes_{v \in T} |\eta_{\nu_T}(v)\rangle\langle\eta_{\nu_T}(v)| \Big) \sigma_i \Big(\frac{\mathsf{I}}{\mathrm{Tr}(\mathsf{I})} \bigotimes_{v \in T} |\eta_{\nu_T}(v)\rangle\langle\eta_{\nu_T}(v)| \bigotimes_{v \in T} |\delta(v)\rangle\langle\delta(v)| \Big) \sigma_j \right)$$

6.5 Full Delegated MPQC Protocol and Security Analysis

We now present the DMPQC Protocol 19 constructing Resource 14 in more detail. Without loss of generality, we suppose that the computation accepts input qubits in the order defined by the ordering of Clients and outputs qubits in that same order and that all Clients have the same input and output size of one qubit. The protocol can easily be extended to Clients with multiple input and output qubits by

applying to each input and output the exact same steps as for the single input and output used here. We choose this presentation for simplicity's sake only, thus avoiding the introduction of additional indices. Furthermore, as mentioned in Section 6.2, we assume that the base graph chosen by the Clients for performing the initial MBQC computation of the joint unitary has degree 1 on all input vertices of set I.

Before proving the security of our full protocol, we first prove the composable security of the VBQC Protocol as captured by comparing it with the Verifiable Blind Delegated Quantum Computation Resource 12 from [41] (Lemma 6.4). We use for this the Local-Criteria Reduction Theorem 3.4. This is required since up to now only local criteria were used in defining its security.

Lemma 6.4 (Composable Security of VBQC)**.** *If the VBQC Protocol is ϵ_V verifiable, then it ϵ'_V-statistically-constructs the VDQC Resource 12 from Insecure Quantum Channels (Ideal Resource 4), for $\epsilon'_V = 2^{2N+2+1/2}\sqrt{\epsilon_V}$, where N is number of the Client's input qubits.*

Proof. We need to show that the VBQC Protocol satisfies the local criteria described in [41] to use their reduction result, restated in this thesis as Theorem 3.4. We recall the four local criteria: correctness, blindness, verifiability and input-independence of verification. The first three are proven in [78] (as stated in Sections 3.5.1 and 3.5.2) and the last one is satisfied if the Adversary can deduce on its own whether the verification has succeeded or not without this information affecting the blindness of the protocol.

We have modified the original protocol of [78] by making the Client announce at the end whether it has accepted or rejected the outcome of the computation and show that it remains secure. In this case, the input-independence is trivially satisfied but we must show that this has not compromised the blindness (the verification is not affected). If the Client has not aborted due to the Server's attack, then it has only affected non-trap qubits. This is equivalent to an attack on the UBQC Protocol which is perfectly blind, meaning that this attack does not reveal any information about the Client's input. On the other hand, if the Client aborts, then the Server's attack has affected at least one trap qubit. This gives at most one bit of information about the state of trap qubits, which is independent of the input or the computation. This shows that the protocol with announced verification result remains blind, which in turn concludes the proof.

■

Note that the size of the quantum input impacts the security bound through this reduction. This could potentially be avoided by reproving directly the security of the VBQC Protocol in AC. However, since ϵ_V can be made arbitrarily small (it is negligible in the security parameter independently of the number of parties), the only influence is on the choice of this parameter which needs to be adapted to take into account this degradation of security.

We now prove that Protocol 19 statistically-constructs the MPQC Resource 14 from the Classical SMPC Ideal Resource, Authenticated Classical Channels between all parties and Insecure Quantum Channels between each Client and the Server (Theorem 6.3). We prove correctness and then security separately. This proof uses three other results. The first one is the security of the VBQC Protocol as emulation of the Verifiable Delegated Quantum Computation Ideal Resource from Lemma 6.4 above. The next one corresponds to the security of the DBQC Protocol, which is proven in Section 6.3. Finally

[15] See Protocol 19: if instead the output is classical, the Server simply measures all qubits, sends the results to the Classical SMPC, that either aborts or sends the decrypted outputs to the Clients depending on the trap verification procedure.

Protocol 19 Delegated Multi-Party Quantum Computation

Public Information: Description of a graph $G = (V, E, I, O)$ with input vertices I and output vertices O and a partial ordering over vertices. This corresponds to leak l_ρ.
Inputs:

- The N Clients have collaboratively chosen the description of unitary U acting on $M = \#I$ qubits as an MBQC measurement pattern over graph G (consisting of measurement angles $\{\phi(v)\}_{v\in O^c}$ for non-output qubits in the graph and a flow).
- Each honest Client $j \in [N]$ has a quantum register $\mathcal{X}_j$ which contains their respective part of a collectively possessed state ρ_{inp}.
- The Server has no input.

Outputs: Each Client receives either a state corresponding either to its part in the correct output state $U \cdot \rho$ or Abort.
Protocol:

1. Each Client j chooses an index $p_j \in_R \{1, 2, 3\}$, indicating in which position in its input base-location it will send its input qubit. It sends this value to the Classical SMPC. For its input base-location, it prepares two $|+\rangle$ states for positions $\{1, 2, 3\}\backslash p_j$. Client N also prepares states $|+\rangle$ for non-input qubits of the Dotted-Triple Graph. The Classical SMPC chooses uniformly at random a colouring of the Dotted-Triple Graph compatible with these input positions and deduces from it the unitary applying Hadamard gates for dummy qubits.
2. The Clients and the Server perform the Double-Blind Quantum Computation Protocol 18 with the Classical SMPC as the Orchestrator, which uses this transformation above as the DBQC unitary.
3. The Server entangles using CZ gates the qubits obtained at the end of DBQC Protocol according to the edges of the Dotted-Triple Graph $DT(G)$.
4. The Classical SMPC instructs the Server to perform bridge operations on each qubit in the base-locations of the edges linked to the input layer by measuring them according to the following angles:
   - Traps and dummies are measured as in the VBQC Protocol (random angle for dummies, correct angle for traps up to a random π rotation).
   - The computation qubit is used as a bridge and the angle δ sent to the Server uses the base angle $\phi = \pi/2$. Let b be the measurement result.
   - The measurement angles of the primary computational qubits in the two base locations linked to this edges are updated by adding $(-1)^{1+b}\pi/2$ to the associated θ.
5. The Classical SMPC instructs the Server to measure each remaining qubit $v \in O^c$ with angle $\delta(v)$ computed according to the VBQC Protocol. The Server measures the qubit v in the basis $\left\{\left|+_{\delta(v)}\right\rangle, \left|-_{\delta(v)}\right\rangle\right\}$ and returns the measurement result $b(v)$ to the Classical SMPC.
6. Quantum Output Key-Release Procedure:[15]
   a) The Server sends to each Client the three qubits from the Dotted-Triple-Graph that correspond to their output base-locations.
   b) Each Client receives from the Classical SMPC the position of the computation, trap and dummy qubit and measurement angle $\delta(v)$ for the trap.
   c) Each Client measures its output trap qubit in the basis $\left\{\left|+_{\delta(v)}\right\rangle, \left|-_{\delta(v)}\right\rangle\right\}$ and sends the result to the Classical SMPC.
   d) The Classical SMPC verifies that all traps have been measured correctly using the same verification procedure as in the VBQC protocol (it know all the measurement results, as well as the secret parameters $r(v)$ and the value of the neighbouring dummies of each trap). It sends Abort to all Clients if it fails.
   e) Otherwise, the Classical SMPC sends to each Clients the Q-OTP key for the output computation qubit.
   f) Each Client undoes the Q-OTP on its output computation qubit and sets this quantum register as its output.

we will use the fact that, for the choice of unitary transformations implemented by the DBQC Protocol to prepare the VBQC client-encrypted state, any deviation during its execution can be commuted to the end in a way that does not depend on the secret parameters of the honest Clients (Theorem 6.2 and Lemma 6.3, proven in Section 6.4).

Theorem 6.3 (Composable Security of the DMPQC Protocol)**.** *If the VBQC Protocol of [78] ϵ'_V-statistically-constructs the Verifiable Blind Delegated Quantum Computation Ideal Resource 12 and the DBQC Protocol 18 ϵ_D-statistically-constructs the Double-Blind Quantum Computation Ideal Resource 20 from Insecure Quantum Channels, then the DMPQC Protocol 19 $(\epsilon'_V + \epsilon_D)$-statistically-constructs the Multi-Party Quantum Computation Ideal Resource 14 from Insecure Quantum Channels and a Classical SMPC Ideal Resource.*

Correctness We consider here that all the parties are honest and prove that the protocol is in that case equal to the Ideal Resource (the correctness error is 0). The correctness of the DBQC Protocol and the Classical SMPC that acts as the Orchestrator mean that the Server's state after step 2 in the DMPQC Protocol is a correct VBQC client-encrypted state. The instructions in the last steps of the DMPQC Protocol correspond to an execution of the VBQC Protocol on this state, driven by the Classical SMPC that performs the same steps as an honest VBQC Client. The outcome is therefore also correct per the correctness of VBQC.

■

Security This proof will not construct explicit simulators for all parties since we will directly prove through a series of hybrid reductions that the DMPQC Protocol is equivalent to the MPQC Ideal Resource using previously proven results. This Resource acts as a black-box that takes as input the Clients' input states and outputs either an abort message or the correct output to all parties (with malicious parties receiving the output first). In the Protocol on the other hand, the parties send additional states to construct the graph state collaboratively and communicate classically, which therefore need to be removed. The proof is driven by the goal of applying the security result for the VBQC Protocol and we must therefore indistinguishably transform the state preparation and key release steps into ones resembling those of the VBQC Protocol.

Using Results for DBQC to Simplify State Preparation. We must first transform the output state of the DBQC Protocol into an honestly-generated VBQC client-encrypted state. To this effect, we use the specific properties of our DBQC Protocol to rewrite the potentially deviated output state in a way that separates the state preparation and the deviation.

We remark that we can at will choose to separate the Server (and Classical SMPC) into two entities S_1 and S_2 (respectively T_1 and T_2), so long as we authorise them to share state (quantum for the Server, classical for the SMPC). S_1 act during the DBQC Protocol while S_2 receives the output of the DBQC Protocol and uses it in the latter steps of the DMPQC Protocol. The Classical SMPC T_1 performs the same actions as an honest Orchestrator during the DBQC Protocol, while T_2 coordinates the VBQC execution and key-release steps.

The Adversarial Clients and S_1 may deviate during the DBQC Protocol. However Lemma 6.3 states that, up to a relabelling of the Server's deviation, there is then a one-to-one correspondence between the state at the end of an execution of the DBQC Protocol followed by VBQC and that after a VBQC

execution on an honestly-generated VBQC state. It is therefore possible to rewrite the DBQC output state $\mathsf{QOTP}_k \circ \mathcal{E}(\rho_{H,M,U})$ obtained by S_2 at the end of the protocol[16] together with the instructions $\delta(v)$ given by the Classical SMPC T_2 as $\mathcal{E}'(\rho_V \otimes \rho_{aux})$. There ρ_V is a state obtained by the Server from an honest Client during an execution of the single-client VBQC Protocol upon which the Server performs an arbitrary deviation independent of the secret parameters that were used to generate this state. This a purely formal rewriting procedure and therefore it is indistinguishable from the original protocol.

Using the composition property of the AC framework, the DBQC Protocol can be replaced by the DBQC Ideal Resource 20. Since both are perfectly indistinguishable so long as the Orchestrator is honest (and here it is instantiated with the Classical SMPC), the output state is the same as above. This yields the hybrid presented in Reduction 2, the distinguishing advantage with the DMPQC Protocol is ϵ_D as stated in Theorem 6.1.

Reduction 2 Replacing DBQC Protocol with Resource

1. The Clients and Classical SMPC perform the same initialisation procedure as in the protocol: they place their input qubit at random in their input base-location and prepare $|+\rangle$ states for the other qubits that they own. These are all encrypted and the position and encryptions are sent to the Classical SMPC T_1, who then chooses at random the unitary to be applied (according to its random choice of Dotted Triple-Graph).
2. The Clients send their qubits to be transformed into a VBQC client-encrypted state to the DBQC Resource, the Classical SMPC T_1 sends the unitary as the Orchestrator and S_1 can choose an input state ρ_{aux} together with a malicious coalition of Clients and deviation $\mathcal{E}'$. S_2 recovers the output state $\mathcal{E}'(\rho_V \otimes \rho_{aux})$ as defined in Resource 20, along with a state ρ_S from S_1. The Classical SMPC T_2 recovers the key k and a classical message **state** from T_1 containing the description of the random parameters of the Dotted Triple-Graph.
3. S_2 follows the instructions given by the Classical SMPC T_2 according to the VBQC Protocol for non-output qubits.
4. The Clients and S_2 perform the Quantum Output Key-Release Procedure along with the Classical SMPC T_2.

Rearranging of Honest and Malicious Behaviour during VBQC State Generation. In all generality we can then assume that the deviation occurs after DBQC has been performed: equivalently, S_2 receives an honestly prepared VBQC client-encrypted state ρ_V from the DBQC Ideal Resource, along with the deviation $\mathcal{E}'$ and state ρ_{aux} from S_1 and applies it to ρ_V. Note that in that context, no Clients are malicious since all the malicious behaviour is contained in this deviation and auxiliary state (up to deviations on their input state as explained in Lemma 6.3, which is always allowed).

The effect is that the Clients in this new hybrid only send correct states to the Resource to be transformed into a VBQC client-encrypted state, along with their inputs (i.e. they no longer need to send additional state to be used in the DBQC Protocol). Since these state are honestly prepared, it is equivalent to having a trusted third party (modelled as a Resource), which we call the *input Super-Client* which receives the inputs from the Clients, encrypts them and prepares the rest of the input state to the DBQC Resource by itself.

[16] And therefore also the one sent by the Ideal Resource since the proof of security of DBQC show that the deviation can be replicated by the Simulator in the ideal execution.

Note that since the deviation on the malicious Client's inputs has been incorporated as an (adversarially known) modification of their inputs, the only remaining deviation happens after the Server S_2 has received the output state from the DBQC Resource. Therefore the interface of the Server on the DBQC Resource is now filtered. This results in the following hybrid in Reduction 3. The distinguishing advantage compared to the previous hybrid is 0.

Reduction 3 Transferring Honest Operations to Super-Client

1. The Clients send their input qubits to the input Super-Client, which then encrypts them, positions them and prepares additional $|+\rangle$ states. These parameters are sent to T_1, who chooses the unitary to be applied as previously.
2. The input Super Client sends its qubits to the DBQC Resource, the Classical SMPC sends the unitary as the Orchestrator. S_1 does not input a deviation in the DBQC Resource but sends $\mathcal{E}'$ and state ρ_{aux} to S_2, which furthermore receives the output of the DBQC Resource ρ_V, an honestly generated VBQC client-encrypted state. The Classical SMPC T_2 receives a classical message **state** from T_1, which contains the secret parameters of ρ_V.
3. S_2 applies $\mathcal{E}'$ to $\rho_V \otimes \rho_{aux}$ and proceeds as above.

We notice that the operations performed by S_1 consist only of internal operations, therefore it can be dropped in favour of subsuming these steps into S_2 (i.e. the choice of deviation $\mathcal{E}'$ and the state ρ_{aux} can be made by S_2 directly). Furthermore, we merge the DBQC Resource with the Classical SMPC T_1 and the input Super Client since they are both trusted third parties (and therefore Resources) as well.[17] The result is simply a new Resource which we call Collaborative VBQC State Preparation Resource, that takes as input the input qubits of all Clients, prepares the VBQC client-encrypted state ρ_V and sends it to the Server. There is one additional interface that outputs the description of the randomness associated with this state, which is plugged into the Classical SMPC T_2. This results in Reduction 4.

Reduction 4 Merging Input Resources and Adversaries

1. The Clients send their input qubits to the Collaborative VBQC State Preparation Resource, who outputs ρ_V to Server S_2 and the description of the secret parameters to the Classical SMPC T_2.
2. The Classical SMPC T_2 instructs the Server to measure the non-output qubits according to the VBQC Protocol. At the end, S_2 sends the qubits of the output layer to the appropriate Clients
3. The Classical SMPC T_2 perform the key-release step as in the DMPQC Protocol.

Simplifying Output Recovery Procedure. At this point, up to the quantum key-release step, the execution between the Server, the Collaborative VBQC State Preparation Resource and the Classical SMPC T_2 is exactly the same as an execution of the VBQC Protocol. We now focus on this final step, which consists of the following: (i) each Client measures the trap in their output base-location according to the instruction provided by the Classical SMPC and returns the outcome; (ii) the Classical SMPC T_2 either sends Abort to the Clients and the Server or sends Q-OTP keys to the Clients; (iii) if there is no abort, each Client decrypts its output.

[17] Since Resources are described as CPTP maps, if an interface from one Resource is linked to another Resource they can be merged simply by composing the CPTP maps corresponding to these interfaces.

Any deviation by the malicious Clients after receiving the qubits from their output base-location either lead to an abort (if they affect a trap) or correspond to a deviation on their outputs. Because, in the ideal execution, the malicious Clients are allowed to deviate arbitrarily on their outputs, deviations on output qubits for non-ideal executions do not correspond to attacks as they can be mapped to actions performed after the protocol has concluded. In the absence of abort, all the trap measurement outcomes are correct and we can conclude that the output state is correct up to allowed deviations on the output qubits of malicious Clients. If the deviation affects traps however, the Classical SMPC aborts similarly to an honest Client in the VBQC Protocol. This is then equivalent, up to a rewriting of the malicious Clients' deviations to include them as a deviation due to the Server S_2 during the VBQC Protocol execution, to having an *output Super Client* recovering the output layer from the Server, performing the measurements on traps as instructed by the Classical SMPC T_2 and, if there is no abort, decrypting the output before sending to each individual Client its own output state. This results in Reduction 5.

Reduction 5 Recovering an Honest-Client VBQC Protocol

1. The Clients send their input qubits to the Collaborative VBQC State Preparation Resource, who outputs ρ_V to the Server and the description of the secret parameters to the Classical SMPC T_2.
2. The Classical SMPC T_2 instructs the Server to measure the non-output qubits according to the VBQC Protocol. At the end, the Server sends the output qubits to the output Super Client.
3. The Classical SMPC T_2 instructs the output Super Client to measure the trap qubits. If a trap has been incorrectly measured, T_2 outputs Abort to the output Super Client, who forwards it to the Server and the Clients.
4. Otherwise, T_2 sends the decryption keys to the output Super Client, who then decrypts the output and sends to each Client its output.

Application of VBQC Security Result. We again use the Resource fusion property to merge the Collaborative VBQC State Preparation Resource, Classical SMPC T_2 and the output Super-Client. This new Collaborative VBQC Client Resource performs exactly the same steps as an honest VBQC Client using the collaborative input state provided by the individual Clients, after which it distributes the output states back to the individual Clients.

Finally, we use the result of Lemma 6.4 which shows the equivalence between the VBQC Protocol with an honest Client and the Verifiable Delegated Quantum Computation Resource 12. In fact, the Server only ever interacts in the hybrid above with honest parties (or equivalently, Resources). It is therefore possible to replace the VBQC Protocol execution between the Collaborative VBQC Client Resource and the Server with the VDQC Resource, at a cost of ϵ'_V.

After this change, the Collaborative VBQC Client Resource is replaced by one which simply forwards the inputs of the individual Clients to the VDQC Resource, and later recovers the output and distributes it to the individual Clients (or aborts). These three Resources – input aggregation, VDQC and output distribution – can then be further merged to exactly form the MPQC Ideal Resource 14. It is furthermore statistically equivalent, up to a combined cost of $\epsilon_D + \epsilon'_V$, to the initial DMPQC Protocol. This concludes the proof.

■

We finally note that, since the DBQC Protocol perfectly emulates the DBQC Ideal Resource, the DMPQC Protocol ϵ'_V-emulates the MPQC Ideal Resource.

6.6 Implementing the Classical SMPC Resource

We now describe an implementation of the Classical SMPC Resource 10 used by the parties in the DMPQC Protocol 19 by levaraging the Universal Thresholdiser construction presented in Section 3.5.4. This demonstrates the simplicity of the classical functions that are required during the execution of the protocol. We first define the functions that are needed to implement the various step in the protocol which require a Classical SMPC. Then, we briefly present how they are used in the protocol to implement the various actions of this resource. The threshold of the UT is set to $t = N$, meaning that all players must collaborate to decrypt the result of the computations. Recall that applying Eval only produces a partial evaluation which cannot be decrypted unless Combine is called on t such partial evaluations. To avoid repetitions, we often directly say which function is applied at each step without mentioning necessarily Eval. However, the only decrypted values are the ones where Combine is explicitly called. We further assume that Verif is called on the partial evaluations before any call to Combine, thus verifying that all these values have been properly computed.

6.6.1 Useful Functions

We will be using the UT in the following way: the creation and sharing of secret parameters through the Setup protocol uses the Secure UT Key Setup Resource 15, while the other function (Eval, Verif and Combine) will be left to the participants of the protocol (as they are composed of local operation and broadcasts). The functions that need to be evaluated are presented here in order of appearance in the protocols presented in the next subsection. The total depth of the functions applied is independent of the size of the computation and we suppose that it is upper-bounded by a constant d. Note that all operations are either operation in $\mathbb{Z}_8$ or $\mathbb{Z}_2$.

1. $\texttt{XOR}(\{\mathbf{str}_j\}_{j\in[N]}) \to \mathbf{str}$ takes as input N binary strings $\{\mathbf{str}_j\}_{j\in[N]}$ and outputs their bitwise XOR.
2. $\texttt{Gen-DTG}(G, \mathbf{list}_{inp}, \mathbf{rand}) \to (\mathbf{list}_{DTG}, \mathbf{list}_{out})$ takes as input the description of a graph $G = (V, E, I, O)$, a list $\mathbf{list}_{inp}$ indicating for each input the position of the computation qubits in its base-location (where the actual input will be placed), and a random value $\mathbf{rand}$. The list $\mathbf{list}_{DTG}$ specifies the nature of each qubit in the Dotted-Triple Graph in a known public order: if the v^{th} entry in the list is 0 (respectively 1), qubit v in the Dotted-Triple Graph is a rotated (respectively dummy) qubit. The list $\mathbf{list}_{out}$ specifies the nature (computation, trap or dummy) of the output qubits. Note that the local nature of the DTG construction ensures that this function is independent of the depth of the computation (i.e. it can be parallelised).
3. $\texttt{U-to-MBQC}(U) \to \{\alpha(v)\}_{v\in O^c}$ takes as input a valid description of a unitary (as a sequence of gates) and outputs a sequence of MBQC measurement angles $\{\alpha(v)\}_{v\in O^c}$ implementing the unitary on a graph $G = (V, E, I, O)$ and with flow $(f, \preceq)$ (the graph and flow are hard-coded in the function and we suppose that this function is only called on unitaries that can be implemented on it). This function is used on very specific unitaries and can therefore be heavily optimised:

it simply chooses between two possible sets of angles (for dummy insertion, one set computing Hadamard and the other Identity), which only differ on one angle (0 for dummies and $-\pi/2$ for the others). It can therefore be simplified to only updating this angle if it is required.

4. $\texttt{State}-\texttt{Init}(\{\theta_j(v)\}_j, \mathbf{state}_i, \mathbf{t}) \rightarrow \theta$ takes as input values $\{\theta_j(v)\}_j$ and measurement results $\mathbf{t}$ and outputs the value $\theta(v)$ of the unmeasured qubit's angle at the end of the State Preparation Protocol. The optional value $\mathbf{state}_i$ is used only in the case of the Quantum One-Time-Padded input qubit of player i to the DBQC Protocol since is contains the value of the encryption angle of the owner's input.
5. $\texttt{Updt}-\texttt{Enc}(\mathbf{state}, \mathbf{enc}, \mathbf{list}_{DTG}) \rightarrow (\widetilde{\mathbf{state}}, \widetilde{\mathbf{enc}})$ implements the discussion of Section 6.4.1.1, taking as input a list **state** containing secret parameters describing the state of qubits (or encryption keys for inputs), a list of encryption keys and a list $\mathbf{list}_{DTG}$ indicating the nature of each qubit in the Dotted-Triple Graph. It outputs updated values for the state and keys as follows. For dummy positions, the value of list **enc** is 0 and the function exchanges this value with the one at the same position from **state**. For traps and non-input computation qubits, **state** contains a value θ and **enc** contains values $\tilde{\theta}$ and $\tilde{a}$ and, after application, **state** contains $\tilde{\theta}$ and **enc** contains values $\theta - (-1)^{\tilde{a}}\tilde{\theta}$ and $\tilde{a}$. For inputs, **state** contains values θ and a and **enc** contains values $\tilde{\theta}$ and $\tilde{a}$ and, after application, **state** contains $(-1)^{a\oplus\tilde{a}}\tilde{\theta}$ and $\tilde{a}$ and **enc** contains values $\theta - \theta'$ and $a \oplus \tilde{a}$.
6. $\texttt{Ang}_{UBQC}(G, (f, \preceq), v, \phi(v), \theta(v), \mathbf{r}, \mathbf{b}) \rightarrow \delta(v)$ takes as input a UBQC graph G and its flow $(f, \preceq)$, the index of a non-output qubit v in the graph G, the original measurement angle defined by the computation $\phi(v)$, the secret parameters $(\theta(v), \mathbf{r})$ and previous measurement results $\mathbf{b}$ and outputs the corrected measurement angle $\delta(v)$. The function $\texttt{Ang}_{VBQC}$ performs the same computation but taking $\mathbf{list}_{DTG}$ as parameter as well (the position of the dummies and traps, we suppose that for non-trap qubits in the output layer, this function always returns angle 0). The angle update functions are the ones described in the Section 2.3.
7. $\texttt{Apply}-\texttt{H}(\widetilde{\mathbf{state}}, \mathbf{list}_{DTG}) \rightarrow \mathbf{state}$ takes as input a list of secret parameters describing the state of qubits (or encryption keys for inputs) and a list $\mathbf{list}_{DTG}$ indicating the nature of each qubit in the Dotted-Triple Graph and outputs **state** in which, each qubit position q such that $\widetilde{\mathbf{list}_{DTG}}[q] = 1$ (it is a dummy) contains (the classical description of) $|0\rangle$ and the rest is identical to $\widetilde{\mathbf{state}}$.
8. $\texttt{Key}_{UBQC}(\mathbf{r}, \mathbf{b}, v, (f, \preceq)) \rightarrow k_q$ takes as input the secret parameters $\mathbf{r}$ and measurement results $\mathbf{b}$ and outputs the key k_v for the Quantum One-Time-Pad on qubit v in the result of the UBQC computation whose flow is given by $(f, \preceq)$. $\texttt{Key}_{VBQC}(\mathbf{r}, \mathbf{b}, v, (f, \preceq), \mathbf{list}_{DTG}) \rightarrow k_v$ performs the same but takes as input $\mathbf{list}_{DTG}$ since the keys are only for computation qubits (here v corresponds to an output *base-location*).
9. $\texttt{Apply}-\texttt{OTP}(\widetilde{\mathbf{list}}, \mathbf{OTP}) \rightarrow \mathbf{list}$ takes as input a list of secret parameters for a VBQC scheme and the key of a Quantum One-Time-Pad and outputs the updated secret parameters resulting from the application of this One-Time-Pad to the qubits corresponding to these secret parameters.
10. $\texttt{Str}-\texttt{Extr}(\mathbf{list}, O) \rightarrow \mathbf{ind}$ takes as input a list **list** and indices O in this list and outputs the sub-list **list** corresponding to those indices.
11. $\texttt{Ver}(\mathbf{r}, \mathbf{b}, \mathbf{list}, val) \rightarrow \texttt{ok} \in \{0, 1\}$ takes as input three lists and a value and outputs 0 if, for all i such that $\mathbf{list}_i = val$, $\mathbf{r}_i = \mathbf{b}_i$ (and 1 otherwise, indicating a failed verification).

6.6.2 Constructing the Classical SMPC Functionalities

We can now present the way that the UT is used in the protocol to replace the Classical SMPC. It is important to recall that the value of a cypher-text (and partial evaluation) remains hidden unless all players decide to run the protocol Combine on published ciphertexts. As mentioned above, the partial evaluations can be reused in other functions if the total depth of the circuit computing the composed function does not exceed the total depth bound of the UT scheme defined during its setup protocol. This is computationally indistinguishable from the case where there is a Classical SMPC Ideal Resource, as per the UT's security Definition 3.21, so long as there is at least one honest Client.

The Classical SMPC is used in three different steps of Protocol 19. First if generates the Dotted-Triple Graph description and all associated parameters. Then it replaces the Trusted Orchestrator in the DBQC protocol to ensure that the permutation applied to the qubits is remains hidden from the Clients and the Server and therefore that the construction of the Dotted-Triple Graph is unknown as well. Later, it is used to drive the computation on this VBQC graph and, at the end, verify that the traps have been measured correctly and handing out the keys to all Clients if they have.

6.6.2.1 Initialisation UT

All of the parameters from the Dotted-Triple Graph and the associated VBQC computation should be hidden from the Clients and the Server. This initialisation step does not output any value, it simply prepares the values required for the two other roles of the SMPC (it prepares their internal state).

Each Client j knows in which registers they have inserted their inputs, expressed as $inp_j \in \{1, 2, 3\}$ indicating its position in its input base-location in the future DTG. It also knows the Quantum One-Time Pad values (one bit and one angle) that it has chosen to encrypt this input qubit, contained in a list $\mathbf{state}_j$. This first step initialisation step, described in Protocol 20, ends after generating the colouring of the Dotted-Triple Graph.

Protocol 20 UT Implementation of Secret Parameter Generation

Each Client j proceeds as follows:

1. It sends the dummy input λ to the Secure UT Key Setup Resource 15 with constant depth bound d and receives $(pk, c_{\mathbf{p}}, sk_j)$, where $c_{\mathbf{p}}$ corresponds to an encryption under public key pk of the structure of the MBQC computation (graph, flow and measurement angles) that the Clients want to perform on their inputs.
2. It generates encryptions of each angle and bit that it chose to encrypt its input qubit ($\mathbf{state}_j$) and the position of this input in its input base-location inp_j, and encrypts a random value $\mathbf{rand}_{DTG,j}$ (of the format required by the function Gen−DTG). It broadcasts these encryptions to all other Clients.
3. It receives corresponding encryptions from the other Clients and uses Eval to apply the function XOR to generate a partial evaluation of $\mathbf{rand}_{DTG}$ (the XOR of the corresponding bit-strings received from all Clients).
4. It uses Eval to apply Gen−DTG to the encrypted description of the graph, input list corresponding to the concatenation of all encrypted values inp_j supplied by clients, and randomness $\mathbf{rand}_{DTG}$. It recovers partial evaluations of $\mathbf{list}_{DTG}$ and $\mathbf{list}_{out}$.

6.6.2.2 Trusted Orchestrator UT

The Orchestrator from Protocol 18 is required to be honest for it to construct the DBQC Resource 20 transforming a subset of qubits into dummies. In Protocol 19, there is no such party and a Classical SMPC is used instead. We show here how to implement it using UT. We suppose that the Clients have performed the steps in Protocol 20. Let U_D be the dummy insertion unitary defined implicitly by the list $\mathbf{list}_{DTG}$ (if the v^{th} entry in the list is 1, qubit v must be turned into a dummy using H, U_D applies the identity on the other qubits). The following Protocol 21 describes the classical actions of the players during the DBQC Protocol 18.

Protocol 21 UT Implementation of DBQC Orchestrator

Each Client j proceeds as follows:

1. It uses Eval to apply $\mathtt{U-to-MBQC}$ to U_D and get a partial evaluation of MBQC computation angles $\{\alpha(v)\}_{v\in O^c_{U_D}}$ on graph $G_{U_D} = (V_{U_D}, E_{U_D}, I_{U_D}, O_{U_D})$ applying U_D to $3\#V + 9\#E$ qubits.
2. It shares to other Clients (via encryption and broadcast) the random values for $\theta_j(v)$ for measured qubits $v \in O^c_{U_D}$ (apart from its own input qubit whose encryption was shared in the initialisation) used in the State Preparation for the DBQC Protocol. For all qubits $v \in O^c_{U_D}$ it also shares a bit $r_j(v)$. It samples $2(\#V + \#E)$ values $\tilde{\theta}_j(v)$ and $\tilde{a}_j(v)$ for all non-dummy qubits in the future DTG and shares them. These last two parameters are used to implement the basis change discussed in Section 6.4.1.1.
3. It receives from the Server an encryption of the measurement outcomes $t_j(v)$ from the State Preparation phase of the DBQC Protocol.
4. To compute the common random angles $\theta(v)$, it runs the function $\mathtt{State-Init}$ on the encryptions of $\theta_j(v)$, measurement outputs and input encryptions. It also runs $\mathtt{XOR}$ on all $r_j(v)$, $\tilde{\theta}_j(v)$ and $\tilde{a}_j(v)$ to get $r(v)$, $\tilde{\theta}(v)$ and $\tilde{a}(v)$. We denote $\mathbf{enc}$ the list comprising the values $\tilde{\theta}(v)$ and $\tilde{a}(v)$.
5. It partitions the values obtained as output to $\mathtt{State-Init}$ in two lists $\mathbf{state}$ and $\mathbf{enc'}$, the first one containing all values for qubits which later form part of the DTG and the other one containing values for the auxiliary qubits used in DBQC. It then runs $\mathtt{Updt-Enc}$ on $\mathbf{state}$, $\mathbf{enc}$ and $\mathbf{list}_{DTG}$ to get the updated state and encryptions $(\widetilde{\mathbf{state}}, \widetilde{\mathbf{enc}})$.
6. During the computation phase of DBQC, to compute the updated measurement angles $\delta(v)$, it runs the function $\mathtt{Ang}_{UBQC}$ on the angles $\alpha(v)$, $\widetilde{\mathbf{enc}}$, $\mathbf{enc'}$ and $r(v)$.[18] These partial evaluations are sent to the Server, who can recombine them using $\mathsf{Combine}$ to recover $\delta(v)$. It receives from the Server encrypted measurement results in return.
7. Each Client uses Eval to apply $\mathtt{Apply-H}$ on the list $\widetilde{\mathbf{state}}$, thus updating the cypher-text containing the secret parameters of the states in the Dotted-Triple Graph according to the transformation applied by the DBQC Protocol. It then uses $\mathtt{Key}_{UBQC}$ on the Server's measurement outcomes, $r(v)$ and $\mathbf{enc}$ to recover the key to all the output qubits of the DBQC Protocol. Finally, it applies $\mathtt{Apply-OTP}$ on the list containing the values of the DTG state and this new Quantum One-Time-Pad key.

These operations have the effect of updating the secret values describing the state of the computation, trap and dummy qubits in the Dotted-Triple Graph, along with the final encryption of the inputs. This is all contained in the updated variable $\widetilde{\mathbf{state}}$. Choosing and sending all the secret parameters in this step can be done at the same time as the previous ones in the Initialisation UT, thus reducing the number of classical communication rounds. Here it is described sequentially simply for sake of coherence.

[18] This is done at the same time for all measured qubits since U_D is described using only Clifford angles (from the set $\{k\pi/2\}_{0\leq k\leq 3}$) and therefore the flow can be chosen as the trivial flow with no immediate corrections. The corrections do affect the final QOTP on the output state.

6.6.2.3 Collaborative VBQC UT

This Classical SMPC reuses the state of the previously called SMPC, after the DBQC protocol has been completed, and performs the VBQC version of the original MBQC computation that the Clients actually wanted to apply on their inputs. We suppose that they have an encryption of the Dotted-Triple Graph $(\widetilde{\mathbf{state}}, \mathbf{list}_{DTG})$ as described earlier in this section, and an encryption of the angles and flow of the base MBQC computation which were generated through the Setup protocol at the very beginning. The Classical SMPC performing the steps of the honest Client in the VBQC Protocol can then can be implemented by having each Client j perform the following Protocol 22.

Protocol 22 UT Implementation of VBQC Computation and Key-Release

Computation phase of the VBQC Protocol: for each non-output qubit layer L in VBQC graph $DT(G)$, it computes the partial evaluation of the updated measurement angles by using Eval on $\mathtt{Ang}_{VBQC}$ with angles $\phi(v)$ for the initial MBQC computation and $\widetilde{\mathbf{state}}$, $\mathbf{list}_{DTG}$. It sends the partial evaluations of $\delta(v)$ for $v \in L$ to the Server (who then runs the algorithm Combine). It receives in return an encryption of the associated measurement results. This is repeated until the output layer is reached and the Server returns to each Client its output base-location qubits.

Output trap testing:

1. It calculates the nature of qubits in the output of all Clients (including itself) by using N times $\mathtt{Str}-\mathtt{Extr}$ on the cypher-text $\mathbf{list}_{out}$, with the indices for each call j' of this function corresponding to the output qubits of Client j'. It sends each partial evaluation to the corresponding Client j' (keeping to itself the partial evaluation for its own output).
2. For each qubit v in the output layer of Client j', each Client j computes the partial evaluation of the measurement angle using the Eval algorithm of the UT on a classical circuit computing the function $\mathtt{Ang}_{VBQC}$ (recall that we suppose that for non-trap qubits in the output layer, this function always returns angle 0). It sends each partial evaluation of $\delta(v_{j'})$ to the corresponding Client j'.
3. It uses the algorithm Combine to recover the position of computation, trap and dummy qubits amongst its own output, along with the measurement angle corresponding to the traps qubit. After measuring the trap qubit in the basis $\left\{\left|+_{\delta(v_j)}\right\rangle\!\!\left\langle+_{\delta(v_j)}\right|, \left|-_{\delta(v_j)}\right\rangle\!\!\left\langle-_{\delta(v_j)}\right|\right\}$, it sends the encryption of the resulting measurement outcome $b_{O,j}$ to all other Clients.

Verification of trap measurement results: it computes a partial evaluation of ok using Ver on the measurement results, secret parameters for the value of the traps and positions of traps and sends the partial evaluation of ok to all Clients. It uses Combine to recombine ok and outputs Abort if it is equal to 1. If it did not receive Abort, it continues.

Output Key-Release: it calculates consecutively for each Client j' the partial evaluation of the OTP key for all qubits in this Client's output layer using $\mathtt{Key}_{VBQC}$ on measurement results and secret parameters of DTG before sending it to Client j'. It uses the algorithm Combine on the values received from other Clients to recover the OTP keys for its own output.

6.7 Performance Analysis and in-depth Comparison with Previous Work

We now compare our result with [40, 90, 5]. Reference [40] also achieve an information-theoretic upgrade of a Classical SMPC to the quantum domain, secure against an arbitrary number of corrupted parties. On the other hand, the protocol from [5] is only computationally-secure since it relies on a Fully-Homomorphic Encryption Scheme on top of the Classical SMPC, but it is also secure against arbitrary corruptions. The protocol of [90] constructs an information-theoretically secure Quantum SMPC but

suffers from an artificial blow-up in the number of participants and exchanged qubits.[19] The protocols of [90, 5] are proven secure in the Stand-Alone Model, whereas ours and that of [40] are fully composable. On top of blindness, all protocols provide verifiability with unanimous abort apart from that of [5] who achieves the stronger notion of identifiable abort.[20]

One key advantage of our protocol over the others lies in its delegated nature, where only one participant needs a full fault-tolerant quantum computer while the rest only perform very limited quantum operations, compared with the symmetric setup in [40, 90] where all participant has requires fault-tolerance. The protocol of [5] can be considered semi-delegated in the sense that the brunt of the quantum computation is performed by a single player. However, all players must have the ability to perform arbitrary Cliffords on large states and cannot do so without having at their disposal a full fault-tolerant quantum computer. This is also reflected in the network topology: whereas the best performance in [40, 90, 5] can only be reached by using a complete quantum and classical communication graph, we only need a star graph for quantum communications.

Regarding classical primitives, [90] only requires secure coin-tossing and authenticated broadcast channels (information-theoretically secure since they can rely on an honest majority). We only use our Classical SMPC to perform coin-tossing, basic string operations (selection in array) and computations in $\mathbb{Z}_8$ and $\mathbb{Z}_2$. The Classical SMPC is more complex in [40, 5] since it must be able to sample uniformly at random and perform computations on the classical descriptions of arbitrary Cliffords.

We can now quantify more precisely the number of classical rounds of communication or calls to the Classical SMPC resource, quantum rounds of communication, and size of quantum memory required by each participant in the protocol. Let N be the number of parties, d the depth of the computation (MBQC for our paper, circuit for [90] and $\{\mathsf{T}, \mathsf{CNOT}\}$-depth for [40]), t the number fo T gates, c the number of CNOT gates and η a statistical security parameter.

The Protocol of [40] calls the Classical SMPC very often: a constant number of times for each input qubit and gate in the circuit. But the most costly part is the generation of ancillary magic states (for implementing T gates via gate-teleportation), which requires $\mathcal{O}(\eta(N+t))$ invocations of the Classical SMPC. Our protocol simply uses $d+5$ calls to this Resource, 2 of which are made for setting up the state and 3 for the key-release step (2 for classical outputs). This is equivalent to the classical communication requirements of [90], where they only need $d+2$ classical broadcasts per participant (one for setting up the shared randomness and another for the state preparation, while the calls during the computation can be parallelised). If all quantum communications are done in parallel in [90], it can be further parallelised to only require a constant number of classical broadcast rounds. The protocol of [5] uses FHE (classical and quantum) to perform the computation and consequently the number of calls to the Classicl SMPC is only constant. We note that using another classical primitive called functional encryption, where a party in possession of an evaluation key can recover the clear-text of a function of the encrypted values (and only that), would allow to attain the same result for our construction by allowing the Server to compute

[19] It is based on error-correcting codes and the size of the code must correspond to the number of players N. The maximum number of cheaters tolerated by the protocol is the number of correctable errors $\left\lfloor \frac{C_{dist}-1}{2} \right\rfloor$, which by the quantum Singleton bound [117] is at most $\left\lfloor \frac{N-1}{4} \right\rfloor$. In their example, 7 players are required for implementing a two-party computation since the code that is used is of size 7 and corrects 1 error. This leads to a situation where 5 participants that don't have inputs nor outputs must still exchange messages and none can be malicious if one of the players with inputs is.

[20] A protocol satisfies the unanimous abort property if all honest players abort at the same time, as compared with selective aborts where the Adversary can choose which players will abort separately. On top of that, identifiable abort means that all honest players agree on the malicious party responsible for the failure of the protocol.

the next measurement angle as a function of the encrypted secrets and previous measurement results.

The protocol of [40] requires numerous rounds of quantum communication as they need to send encoded states around for the verification of inputs and T and CNOT gates. After parallelisation the total cost is $\mathcal{O}(Nd)$ quantum rounds. [5] aims to remove the circuit dependency in the number of rounds, obtaining $\mathcal{O}\big(N^4\big)$ quantum rounds in the worst case in the case where the protocol is parallelised.[21] [90] seeks to optimise the quantum memory requirement of players and therefore their communication is done sequentially, yielding $\mathcal{O}\big(\eta^2(N+t)\big)$ quantum rounds. Parallelisation lowers it to 3 (or 2 for classical outputs), at a higher quantum memory cost for all parties. Our protocol is optimal as there are only 2 quantum rounds (1 for classical outputs): sending to the Server the inputs and all states required for the collaborative state preparation phase and later recovering the output layer qubits from the Server

Finally, the number of qubits required by [40] during the computation phase is $\mathcal{O}(\eta(N+t))$ for each participant (they encode each of their input qubits, ancillae and magic states using $\mathcal{O}(\eta)$ qubits). However they use $\mathcal{O}\big(\eta^2(N+t)\big)$ additional qubits in the offline phase to prepare the ancillary qubits (if the quantum communications are performed in parallel). On the other hand, [90] reduces the number of qubits for each participant to $\mathcal{O}\big(N^2\big)$ for sequential quantum communication, but this blows up to $\mathcal{O}\big(\eta^2 N(N+t)\big)$ if parallelised. The construction from [5] uses a compiler that adds automatically a cost of $\mathcal{O}\big(N^2\big)$ for each base qubit. The costly double encryptions and multiple layers of traps, in particular for the magic state distillation procedure, yields a total quantum memory cost per participant of at least $\mathcal{O}\big(tN^9\eta^2\big)$ (this is a weak lower bound). In our paper the Server needs $\mathcal{O}(\eta Nd)$ qubits to perform the VBQC computation, and it must be able to apply a constant-depth MBQC computation through the DBQC Protocol to these qubits first to transform some input dummies. The graph applied contains 9 qubits per qubit in the final $DT(G)$ used in VBQC. Each qubit in these graphs must be generated using N qubits, resulting in a total qubit cost of $\mathcal{O}\big(\eta N^2 d\big)$ for parallel rounds of quantum communication but only $\mathcal{O}(\eta Nd)$ if they are performed sequentially. However, the Clients can prepare the additional qubits on the fly, therefore each Client only requires three qubits of quantum memory which it uses to store its input at the start of the protocol and the output, dummy and trap at the end. For classical inputs and outputs, the Clients do not even need quantum memory (their inputs can be encoded into the qubits that have been sent by adding π to the rotation angle and the outputs are classical since the Server performs the measurements).

6.8 Conclusion and Discussion

The purpose of any cryptographic protocol is to mimic the ideal scenario where all players send their inputs to a trusted third party and later recover their outputs. This minimal setting implies that at least two rounds of communication are needed even when one assumes that the third participant is indeed honest. In this respect, we achieve an almost-perfect delegated multi-party quantum computation: the quantum communication between all Clients and the Server is round-optimal, while at the same time removing all trust requirements between participants.

[21] They send states along a path of size N^2 in the communication graph of the parties, and remove a party if it doesn't deliver a packet before resending the states along a different path of the same size. In the worst case where there are $N-1$ malicious players which do not want to get caught cheating, they can drop $(N-1)(N-2)/2$ packets before they get disconnected from the communication graph.

This is accomplished through a deconstruction-reconstruction process of a single Client protocol where a new *blind but not verifiable protocol* is introduced to allow multiple Clients to prepare a *good-enough VBQC execution state* collaboratively. This new protocol is highly versatile and can benefit to situations where a unitary must be applied and yet remain unknown to all participants. In our case, by adding straight-forward conditions on the form of the input and the unitary (namely that they do not leak information about the final good-enough VBQC execution state and that the unitary is Clifford), we are able to use the generated state to perform a fully verifiable computation. This depth-independent state preparation step effectively bootstraps the verifiability of any computation by using blindness alone, thus breaking the oft-believed principle that verifiability of the full protocol cannot be achieved unless all sub-components are also verifiable.

Open Questions. Our result leaves a few questions open. The first one is whether it is possible to perform a Multi-Party Quantum Computation with strictly the same number of qubits per Client as the single Client VBQC. Lowering the classical communication requirement to a constant number of rounds is also an interesting problem. Optimal protocols exist in the classical case with only four rounds of communications [65], yet no explicit Quantum Secure Multi-Party Computation Protocol has sub-linear classical round-complexity as of now. Yet another question is whether it is possible to construct a protocol ensuring blindness without verifiability even in the presence of client-server collusion (i.e. extending [76] to arbitrary corruptions). Finally, the Double-Blind Quantum Computation Protocol and Double-Blind Rotated State Preparation that are used to prepare the state for the VBQC computation are interesting implement functionalities never defined before and other protocols could benefit from using these same functionalities as subroutines.

Future Work

We are currently developing an improvement on the State Preparation used to create a VBQC client-encrypted state based on the Double-Blind BB84 State Preparation Protocol. The computation phase in the DBQC Protocol can be drastically simplified in terms of the number of qubits that need to be prepared collaboratively. We can use the bridge/break property of the BB84 state (in an MBQC graph, a state from the computational basis breaks the graph at this vertex, while $|+\rangle$ and $|-\rangle$ create a bridge between two nodes of a graph) to tailor-make a graph specifically for the unitaries that need to be applied (H or I on non-inputs and SWAP or Id on inputs). The difficulty lies in avoiding attacks similar to those presented at the end of Section 6.4.2 which depend on the type of qubits being prepared in the Dotted-Triple Graph.

Another way to reduce the number of qubits sent by each Client would be to use the high similarity between the DBQC implementation of H and I operations. The graphs are identical and all angles but one are the same as well. We are currently in working on a blind quantum computation protocol whose aim is precisely to minimise the number of transmitted qubits in this exact scenario. Preliminary results indicate that qubits need only to be sent for inputs, the vertices where the angles differ and any non-Clifford vertex in their future influence cone, meaning all qubits whose dependencies can be traced back to the hidden vertex whose angles are not in $\{k\pi/2\}$ for $0 \leq k \leq 3$. Since all vertices in our H/I gadget are Clifford, this means that only two qubits would need to be sent per Client and per vertex in the DTG, which is very close to optimal.

We are also looking into more generic ways of proving that a deviation can be commuted through the State Preparation without picking up secret parameters, instead of doing it step-wise in a way that highly depends on the specificities of the protocol. This would help both with the above-mentioned VBQC client-encrypted preparation based on the Double-Blind BB84 State Preparation, but also any other future improvement to this phase of the computation. Looking even further, the definition for good-enough states can be expanded beyond the current one and encompass other verification protocol that require a player to prepare honestly an initial state. The question then becomes that of finding sufficient condition for a State Preparation Protocol preparing these newly defined states to undergo the same commutation procedure. This would expand our bootstrapping technique to other verifiable protocols where a resource state needs to be created and allow them to be turned into multi-party protocols in a similar way.

Finally, both the Double-Blind Rotated State Preparation Protocol and the Double-Blind BB84 State Preparation Protocol share similarities with Secret Sharing Protocols, in the sense that it is possible to see them as the Orchestrator instructing the Clients to send a specific state (instead of the Clients choosing this on their own in the current version). The creation of the collaboratively-generated state then corresponds to the reconstruction phase of the Secret Sharing Protocol. Here the secrets are necessarily classical since the Orchestrator is purely classical. It would be interesting to investigate how other Secret Sharing Schemes where the secrets are efficiently classically describable (in the form of a quantum circuit on known input states, the circuit being as simple as possible) could be used for MPQC. There is a strong parallel with the classical case since Verifiable Secret Sharing is one of the techniques used to secure SMPC Protocols, but in that case it is used without reconstruction unless some malicious parties refuse to cooperate.

Chapter 7

Qubit and Operation Optimal Verifiable Quantum Computations

7.1 Motivation and Overview of Results

7.1.1 Benchmarking and Verification in a Networked Setting

Quantum computing promises unparalleled power for solving certain problems such as database search [64] or integer factoring [124]. Recent experimental progress showed that the limit of classical un-simulatability is now within reach, if not already surpassed [9]. In this regime, quantum computers become so powerful that their classical counterparts cannot simulate their computation in a reasonable time.

On one hand, this has triggered a lot of interest from all stakeholders starting to feel the limitations of classical computing power and looking for ways to circumvent the inevitable slow-down of Moore's law, from academic labs all the way to industry users. This, in turn, has driven most recent algorithmic and software developments in the field. More and more use cases are being studied with the goal of running useful computations on these devices as soon as their capabilities allow it.

On the other hand, because the cloud is emerging as the preferred way of accessing quantum machines, the questions of data and algorithm confidentiality as well as computation integrity are becoming ever more important. First, it is expected that only the most crucial and strategic computations will be run on these quantum computers, thus making these systems ideal targets for sophisticated hacking. Second, disruption caused by (un)intentional mis-computations could remain undetected in the absence of means to check the result. Even after detection, it could still be difficult to pin-point the failing component due to the impossibility of following exactly the progress of quantum computations. Several methods for eschewing this hurdle have been devised in the past (see e.g. [21, 53] and [58] for a review).

However, in spite of these results, the initial questions are far from resolved. This is because currently known verifiable protocols are too sensitive to be of practical use. Indeed, they have been developed for

noiseless devices and have been optimized to detect the smallest fiddling and abort quickly. Unfortunately, replacing perfect devices by even slightly noisy ones is not an option: the verification procedure would keep aborting, mistakenly thinking that plain imperfections are in fact the signature of malicious behaviour.

Several options for dealing with this sensitivity have been discussed in the past. Previous research explored forgoing blindness [57], imposing restrictions on the noise model [71], replacing the server by two entangled but non-communicating servers and classical clients [104], or settling for computational security [96]. Yet these protocols either only achieve inverse-polynomial security bounds or, to obtain exponential security, leverage the full power of fault-tolerant encoding along with a blow-up in the size of the computation space. This makes them impractical for Noisy Intermediate-Scale Quantum devices[1] or even fault-tolerant devices that are able to perform the initial non-verifiable scheme since the fault-tolerant scheme used to boost the verifiability must be used on top of the one that suppresses the machine's inherent noise.

As a consequence, before the wide-spread availability of very large fault-tolerant machines where memory constraints are a non-factor, clients would still be left with no better alternative than to either give up their security objectives entirely or try to convince themselves that providers are not as malicious as they could be. They could start by benchmarking the performance and quality of the available devices (see e.g. [82, 133, 101]) by running computations whose outcomes are known in advance, and decide later based on these results whether to trust or not future runs on the tested service provider. However this strategy falls short of the security expectations of most users because no benchmark entails future fulfilment of the provider's promises: benchmarking is not equivalent to computation verification. It might only serve *a posteriori* to demonstrate that the provider cheated during the benchmark, but would not help preventing a deviant behaviour at the time of computation.

7.1.2 Our Contribution

In this chapter, we propose a solution to these security issues while mitigating the effect of errors by introducing a verifiable, blind and delegated quantum computing protocol for deterministic computations with classical inputs and outputs, that is also robust to noise (Section 7.2.4). It relies on the Measurement-Based Quantum Computation model, presented in Section 2.3, as it is the most natural one for delegation. More precisely, it consists of multiple executions of a blind version of the Client's computation interleaved with test runs aimed at detecting any dishonest behaviour by the Server. These test runs are indistinguishable from the computations. A majority vote over computations is performed at the end.

We show in Section 7.3.1 that any attempt at disturbing the computation will be with high probability either caught or classically mitigated. The associated exponential security bound results from the majority vote which forces the Server to attack at least half the runs in order to have a chance to corrupt the computation, without being able to discriminate between both types of runs due to the overall blindness. We want to stress here that we make *no* assumptions on the adversary in the process. The adversary can be as malicious as it wants and the security will not be compromised. Indeed, our protocol achieves information-theoretic security in the composable framework of Abstract Cryptography (described

[1] These are machines that may be (but are not necessarily) beyond classical simulatability, which starts being impractical above 50 qubits, yet do not have the quantum memory space requirements for full fault-tolerance. Given the engineering difficulties encountered by various teams when trying to scale quantum device, it is widely believed that such limited devices will be the bread and butter of scientists wishing to perform quantum calculation in the foreseeable future [116].

in Section 3.3.2). It ensures it will not be jeopardised by subsequent or simultaneous instantiations in conjunction with other protocols.

On the other hand, the noise-robustness comes both from the classical error-correcting capability offered by majority voting, and by preventing the Client from aborting if only a small number of test runs are triggered. This threshold must be carefully chosen so that the Client still rightfully aborts if there is a risk that the noise will overwhelm the error-correcting capabilities of the repetition code provided by the majority vote, otherwise it could be abused by a malicious server. When analysing the acceptance probability on honest noisy devices, we assume only that the noise is independent across runs (but not necessarily identically distributed) and that the failure probability of a complete test run is lower-bounded by a constant. This error-mitigation property of the protocol means that there is no need to give up the ambition to provide security in a fully malicious adversarial model because of noise: in the presence of honest noise that is sufficiently small, the computation will produce and accept correct outputs with probability exponentially close to 1. Therefore fewer resources need to be spent on the fault-tolerant encoding of the computation since a constant amount of noise per computation will not corrupt our protocol.

The practical implication of verifying the computation by separating dedicated test and computation runs is the absence of overhead for each run used in the protocol when compared to the same non-robust, non-verifiable quantum computation in the MBQC model. In fact, the only overhead of our scheme is the repetition of computations similar to the unprotected one (i.e. same size, connectivities and gate set) a polynomial number of times. In particular, it does not increase the quantum memory requirement nor require additional simultaneous entanglement between quantum systems. To the best of our knowledge, this is the first exponentially-secure delegation protocol that lets the client use the full extent of the available hardware for the computation tasks while tolerating a constant amount of global noise. Any increase in the capabilities of the quantum devices can therefore be used entirely to scale-up the clients' computations. These properties make it the first experimentally scalable global solution to the verification problem, going beyond past experimental feasibility demonstrations of building blocks for verified computations [13, 12, 63, 102] and potentially serving as a blueprint for the development of future quantum cloud applications.

7.2 Building Protocols for Small-Intermediate Scale Quantum Internet, an Iterative Description

The architecture for which our protocols have been developed is composed of two nodes, one acting as a limited Client while the other is a more powerful Server (but only slightly in near-term devices). We suppose that the Client has at its disposal a single qubit on which it can apply a very small number of operations, while the Server has at least two memory qubits (and can perform any computation with high fidelity on them). Any mean of quantum communication can be used to transfer states across these systems, but we describe our protocol based on the one used by trapped ion platforms. We suppose that the qubit of the Client and one of the qubits of the Server can be entangled through a probabilistic process (photon emission, entanglement using a beam-splitter and measurement). We also assume that the Client and the Server can communicate classically through an authenticated classical channel. This setup is summarised in Figure 7.1.

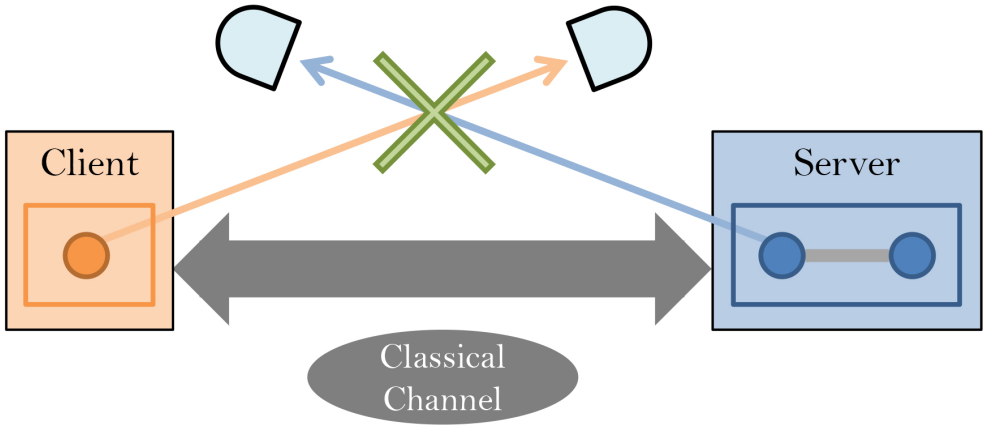


Figure 7.1: Simplest Experimental Setup. The communication qubits of the Client and the Server are probabilistically entangled by making emitted photons go through a beam-splitter. The success of the operation is heralded by their subsequent simultaneous detection in both single-photon detectors.

We will first start by describing the protocol in its simplest form on the minimal hardware presented above, give its equivalent in circuit form and refine it slowly. A weak notion of hiding called flow-ambiguity is applicable to this first protocol. The second protocol ensures that the Server is blind to the computation performed and the Client's input. These will essentially be rewrites of the MBQC and UBQC Protocols presented in Section 2.3 in the context of very limited hardware capabilities. We then further improve the protocol by adding verifiability for the Client against any malicious behaviour by the Server. Allowing the Client to generate states from the computational basis and repeating the blind protocol are the only additional requirements to obtain this verifiability. The final extension in the next Section describes how to improve the protocol in the case where more qubits are available both in the communication and computation phase. This presentation is iterative specifically for the purpose of being as close as possible to the specificities of the experimental setup above. It is meant as a blueprint for future actual proof-of-concept implementations, hence the greater level of detail and apparent repetition.

7.2.1 The Basic MBQC Protocol

For this version of the protocol, it is sufficient to consider that the Server has two quantum registers $\mathcal{C}_{Serv}$ (indexed 1) and $\mathcal{I}_{Serv}$ (indexed 2) containing at least one qubit each. We suppose that the Server can perform a CZ gate between its qubits according to a known graph (not necessarily universal or complete) and measure any of them in the basis $\{|+_\theta\rangle\langle+_\theta|, |-_\theta\rangle\langle-_\theta|\}$ for $\theta \in \Theta$. The protocol that will serve as basis for all others is presented in Protocol 23. We present if for the minimal case where the Server has exactly two qubits.

We do not impose a priori restrictions on the values of angles $\delta(1)$ and $\delta(2)$ (they will depend on the input angles later on). The circuit that is being performed by this protocol is represented by Figure 7.2.

In our case, the graph consists simply of two vertices linked by one edge. Interestingly, this graph has two possible flows. The first one is defined by $I = \{\mathcal{C}_{Serv}\}$ and $O = \{\mathcal{I}_{Serv}\}$ and $f(\mathcal{C}_{Serv}) = \mathcal{I}_{Serv}$, with the partial order then being $\mathcal{C}_{Serv} \preceq \mathcal{I}_{Serv}$, meaning that the outcome of the measurement on $\mathcal{C}_{Serv}$

Protocol 23 Base MBQC Protocol

Client's Inputs: Two angles $(\phi(1), \phi(2))$.
Protocol:

1. The Server initialises the two registers $\mathcal{I}_{Serv}$ and $\mathcal{C}_{Serv}$ in the state $|+\rangle$.
2. The Server performs a CZ gate between the qubits in both of its registers and sends Ack to the Client.
3. The Client sends to the Server a measurement angle $\delta(1)$.
4. The Server performs a measurement in the $\delta(1)$-basis on register $\mathcal{I}_{Serv}$ and sends the outcome of the measurement $s(1)$ to the Client.
5. The Client sends a second measurement angle $\delta(2)$ to the Server.
6. The Server performs a measurement in the $\delta(2)$-basis on register $\mathcal{C}_{Serv}$ and sends outcome $s(2)$ to the Client.

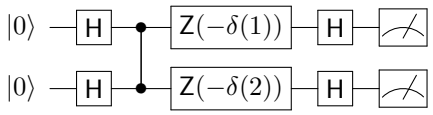
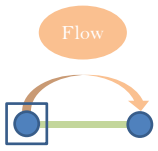


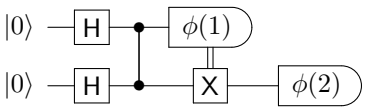
Figure 7.2: Circuit Representation of MBQC Computation Pattern. The final measurement is performed in the computational basis.

will be used to correct the measurement angle on $\mathcal{I}_{Serv}$. The second one is defined by $I = \{\mathcal{C}_{Serv}, \mathcal{I}_{Serv}\}$ and $O = \{\mathcal{C}_{Serv}, \mathcal{I}_{Serv}\}$. In that case a flow would go from the empty set to the empty set, which means that no correction is needed (but we can reuse the same ordering). We call this flow the *trivial flow*. It is important to notice that different flows lead to different computations. These cases are summed up in Figure 7.3 (input qubits are marked with a square). In this case we set $\delta(1) = \phi(1)$ and then $\delta(2)$ can either depend on the measurement outcome (via and X correction) or simply be equal to $\phi(2)$.

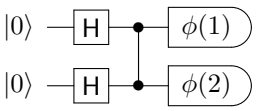


(a) Linear flow from 1 to 2.

(b) Trivial flow.



(c) Correction induced by the linear flow.



(d) No corrections for the trivial flow.

Figure 7.3: Flows with Associated Computation and Correction.

This is the simplest example of a property called *flow-ambiguity*, introduced in [97]. Their main result states that if a graph has more than one flow with the same vertex ordering, then the different computations resulting from using one flow or the other are statistically indistinguishable for the Server. This essentially means that the Client can choose whether to make the angle $\delta(2)$ dependent or not on the

outcome of the first measurement without the Server knowing which one has been chosen. Importantly, the number of flows sharing the same ordering scales with the number of qubits in the computation graph, therefore this property becomes more interesting the more qubits the Server controls (meaning that it is even more useful against powerful Servers). On the other hand, this property is not formulated in a composable framework but using entropy-based definitions. It is unclear how repeating the protocol influences the leakage of information, so we will not consider it in a cryptographic setting.

7.2.2 Upgrading to Full Blindness using UBQC

We now augment the statistical hiding flow-ambiguity of the Base Protocol 23 to full blindness against a malicious Server using UBQC. In contrast with the previous protocol which required no quantum operations on the Client's side but offered a limited variant of hiding, this iteration needs on form of quantum communication or another between the two players so that the Server's initial state is directly or remotely prepared by the Client. For this purpose we first introduce the sub-routine for UBQC State Preparation in Protocol 24. We suppose to that effect that the Client has a quantum register $\mathcal{C}_{Cl}$ as well. The registers $\mathcal{C}_j$ are called the communications registers (of the Client and the Server depending on j) and we suppose that they hold a single qubit each. As explained at the start of this section, these can be probabilistically entangled using a heralded operation which when successful is equivalent to applying CNOT gate between the two registers (with the Client's qubit acting as control). The other register of the Server is called the internal register as it is not similarly accessible.

Protocol 24 UBQC State Preparation

Inputs: The Server's qubit in register $\mathcal{C}_{Serv}$ is initialised in the state $|0\rangle$ and the Client's qubit in register $\mathcal{C}_{Cl}$ is initialised in the state $|+\rangle$.

Protocol:

1. The Client and the Server attempt to establish entanglement on the pair of qubits in register $\mathcal{C}_{Cl}$ and $\mathcal{C}_{Serv}$. The parties thereafter share an EPR-pair in their communication registers after a successful entangling operation.
2. The Client chooses uniformly at random a value $\theta \in_R \Theta$ and performs a measurement in the basis associated to angle $-\theta$ on its register, effectively teleporting the state $|+_\theta\rangle$ up to a Z-correction to the Server's register. The Client records the result of the measurement p and sends Ack to the Server.

Output: The Client outputs (θ, p) and the Server outputs its register $\mathcal{C}_{Serv}$.

The correctness of this state preparation phase follows directly from the rewriting of the EPR-pair using the θ-basis as presented in Section 2.2.3. It is strictly equivalent, in terms of the Server's information gain, to the Client simply preparing the state $|+_{\theta+p\pi}\rangle$ and sending it to the Server.

We now present a UBQC-based Protocol in Protocol 25. Note that the qubits are prepared in the same order as they are measured later during the computation, first the internal qubit and next the communication register. We now suppose that the Client has (in addition to its computation), a classical input x consisting of up to two bits.

Note that all parameters ($\tilde{\theta}(v)$ and $r(v)$) can be chosen by the Client in advance, and the possible values for $\delta(v)$ may be pre-computed easily before the protocol starts: there are only 2 possible values for $\delta(1)$ once $x(1)$, $\tilde{\theta}(1)$ and $r(1)$ are fixed (depending on the value of $p(1)$), and at most 4 values for angle $\delta(2)$ (based on $p(2)$ and $b(1)$).

Protocol 25 UBQC Protocol on Limited Hardware

Client's Inputs:

- Two angles $(\phi(1), \phi(2))$ and a flow f on graph $G = (V = \{1, 2\}; E = \{(1, 2)\})$ describing the computation.
- A classical input to the computation $(x(1), x(2)) \in \{0, 1\}^2$.

Protocol:

1. The Server initialises the registers $\mathcal{C}_{Serv}$ and $\mathcal{I}_{Serv}$ in the state $|0\rangle$ and sends Ack to the Client.
2. The Client initialises its register $\mathcal{C}_{Cl}$ in the state $|+\rangle$ and sends Ack to the Server.
3. They perform one instance of the UBQC State Preparation sub-routine, let $\tilde{\theta}(1)$ be the chosen random angle and $p(1)$ the outcome of the measurement. The Client sets $\theta(1) = \tilde{\theta}(1)+p(1)\pi+x(1)\pi$ and reinitialises its register in the state $|+\rangle$.
4. The Server performs a SWAP gate between its registers $\mathcal{C}_{Serv}$ and $\mathcal{I}_{Serv}$ and sends Ack to the Client.
5. They perform a second instance of the UBQC State Preparation sub-routine, let $\tilde{\theta}(2)$ be the chosen random angle and $p(2)$ the outcome of the measurement. The Client sets $\theta(2) = \tilde{\theta}(2) + p(2)\pi + x(2)\pi$.
6. The Server performs a CZ gate between the qubits in both of its registers and sends Ack to the Client.
7. For $v \in \{1, 2\}$:
   a) The Client chooses at random a value $r(v) \in_R \{0, 1\}$ and sends to the Server a measurement angle $\delta(v)$, defined according to the UBQC angle update Equation 3.16.
   b) The Server performs a measurement in the $\delta(v)$-basis on register $\mathcal{I}_{Serv}$ and sends the outcome of the measurement $b(v)$ to the Client.
8. The Client sets its output as $\{s(v) := b(v) + r(v)\}_{v\in\{1,2\}}$.

It is important to notice that the number of dependencies and hence the number of values to pre-compute only depend on the structure of the graph, with graphs with lower connectivity needing a lower number of pre-computations (as less corrections need to be taken into account). This low number of dependencies was already apparent in Section 4.5.2 for the brickwork states, and is also a low constant for other universal topologies such as cluster-state graphs, and other presently-available non-universal topologies.

This same blinding technique was used previously in protocols that also imposed the computation to be embedded in a universal graph such as brickwork graphs or dotted-complete graphs [78, 53]. This last requirement however caused a blow-up in the number of qubits since the Client could not choose the most optimal graph for its desired computation but had to make it fit these universal graphs. Relaxing this requirement allows us to work directly with the same graph as the one used for the Client's desired computation rather than an expanded one. While this leaks the information about the underlying computation graph, all the other parameters (i.e. computation angles and inputs) remain hidden, which turns out to be sufficient for blindness. The result is a drastic reduction of required memory qubits on the Server's side: so long as the Client's initial computation can be embedded in the Server's architecture, the fully blind version can be implemented as well.

7.2.3 Amplification of Robustness and Verifiability Through Repetition

We can further improve this algorithm by adding verifiability to it through a trappification technique similar to that already presented in Section 3.5.2. The conditions for successfully inserting traps are that they should: (i) have deterministic outcome if measured in the correct basis, (ii) remain undetectable by the Server and (iii) not affect the computation. This last condition – the possibility to still run the initial computation undisturbed – is the most challenging one. Other result modify the base graph in a way that make traps co-exist alongside the computation that is being performed. This results in an additional overhead both in terms of stored qubits, graph connectivity and applied gates compared to the blind version of the computation. We show here how to alleviate this issue.

We start by introducing the Dummy State Preparation as a sub-routine in Protocol 2, similar to the previous one but for the fact that now the Client will measure its register in the computational basis, producing a dummy qubit. The state received by the Server is a copy of the post-measurement dummy state and the Server is unable to distinguish the resulting mixed state from the one it receives in the previous protocol.

Algorithm 2 Dummy State Preparation

Inputs: The Server's qubit in register $\mathcal{C}_{Serv}$ is initialised in the state $|0\rangle$ and the Client's qubit in register $\mathcal{C}_{Cl}$ is initialised in the state $|+\rangle$.
Protocol:

1. The Client and the Server perform an entanglement operation CNOT on registers $\mathcal{C}_{Cl}$ and $\mathcal{C}_{Serv}$.
2. The Client measures its register in the computational basis, receives $p \in \{0, 1\}$ as outcome and sends Ack to the Server.

Output: The Client outputs p, the Server outputs its register $\mathcal{C}_{Serv}$.

The correctness of this state preparation phase follows directly from the CNOT operation. It is strictly equivalent, in terms of the Server's information gain, to the Client simply preparing the state $|p\rangle$ and sending it to the Server. Importantly, the operations from the point of view of the Server are identical to the ones performed in the UBQC State Preparation Protocol 24 above.

Our trap insertion strategies will be to interweave pure computation runs (i.e. without traps) with pure test runs (i.e. containing no computation but only traps). A test run will consist of the Client using the Dummy State Preparation for one of the two State Preparation steps and the UBQC State Preparation for the other. Then, for this run of the protocol, the Server's qubits remain unentangled after application of the CZ gate and a measurement on the non-dummy qubit (henceforth called trap) in the correct basis should always give a deterministic result.

A direct consequence of this construction is that all runs share the same underlying graph G, the same order for the measurements of qubits, and all angles are chosen from the same uniform distribution. We will prove formally later that this implies that the Server cannot distinguish computation and test runs, nor tell which qubits are traps.

Exponential Security Amplification. The above approach to trap insertion is efficient as the only overhead is the repetition of the same sub-protocol. Yet, left as such, it can only achieve a security bound that is inverse-polynomial in the number of runs. For instance, using a single computation run and $n - 1$ test runs would give the Server a $1/n$ chance to corrupt the computation run. The only

previously-known method to reduce the cheating probability to exponentially-low bounds was to merge traps into the single computation run at the expense of using a more complicated graph and then using fault-tolerant quantum error-correction codes on top to achieve the desired amplification of the security (blowing up the overhead in the process). We therefore as of now consider only functions on classical inputs and outputs which are *deterministically computable* in the MBQC framework with angle set Θ.[2] We later prove that this restriction combined with a classical repetition error-correcting code is sufficient to go from inverse polynomial to exponentially-low cheating probabilities.

This capability has two important practical implications. It allows the Client to use the full power of the Server's quantum device to perform its computation in a secure way, whereas previous schemes ate up part of its capabilities to verify the computation. Then, security-wise, even though our protocol remains secure when executed sequentially or in parallel, each call still offers more opportunities for an attacker to succeed.

Note that this amplification technique is common and rather intuitive in purely classical scenarios where attacks can be correlated across various rounds. Although this claim has been made as well in the quantum case in previous works [53, 78, 71], it remained up to now unproven. However, attacks in the quantum realm can be entangled across rounds in a way that is much more powerful than what is possible with classical correlations. This hurdle has not been properly addressed until now and we give the first formal quantum treatment of this technique in the next Section.

Redo Feature. We now described an additional feature that makes the protocol even more suitable for implementation on near-term quantum devices as it vastly improves the probability of successful termination on honest-but-noisy devices. Because the Client or the Server may experience failures in their experimental system, they might wish to discard and redo a given run j. In this case, one of the parties can send a Redo_j request to the other, in which case the parties simply repeat the exact same run albeit with fresh randomness. To prevent the Client from post-selecting on the measurement results returned by the Server,[3] the Redo_j request is allowed only so long as the party asking for it is still supposed to be manipulating the qubits of run j. This is explicitly delineated in the description of the full protocol: the Client cannot ask for a redo after the State Preparations have all been performed but the Server can always make this request.

We show in the next section that this does not impact the blindness nor verifiability of the scheme. This means that a dishonest Server cannot use Redo requests to trick the Client into accepting an incorrect result. This has an important practical impact: without this feature, honest failures of the experimental devices happening during a test run would be counted as a failed test run, thus decreasing drastically the likelihood of successfully completing the protocol. As they can be safely ignored, the only consequence of experimental failures that are caught during the execution is to increase the expected number of repeated runs. This is something that is already used by experimentalists while performing protocols, but we choose to deal with it explicitly in order to prove that this behaviour does not open a loophole in the security of our scheme.

[2] Note that this restriction can be easily loosened to require the computation to be only approximately deterministically computable on the Server's graph for a negligible error probability.

[3] This does not imply that the Client may act maliciously, rather it is meant to explicitly forbid such post-processing by honest Clients in an experimental setup where such optimisations may be tempting.

High-Level Protocol Description. The Client will run the UBQC Protocol n times, but with update rules for the measurement angles that differ depending on the type of run. For $d < n$ runs chosen at random, the Client will update the measurement angles according to the computational measurement pattern, thus resulting in computation runs. The remaining $t := n - d$ runs will be turned into test runs in which the trap qubit is measured in the basis it was prepared in and the dummy is measured in a random basis. In order to tolerate noise in the computation devices, we suppose that a fraction of the traps can be triggered without the Client aborting. Let w be the number of test runs that are tolerated as incorrect. At the end of the protocol, the Client counts the number of test runs where the trap measurement has failed. If this number is higher than w, the Client aborts the protocol by sending the message Abort to the Server.[4] Otherwise it sets the majority outcome of the computation runs as its output and sends message Ok to the Server.

Intuitively, the only way for the Server to disrupt the computation without being detected is by attacking a sufficient number of computation runs (at least $d/2$) without triggering too many test runs. This allows us to boost the verifiability to a value exponentially close to 1. On the other hand, the blindness of the protocol is not affected as it is still perfectly hiding thanks to the composability of the basic UBQC protocol, meaning that repeating it does not break its security.

Full Protocol. The verifiable protocol is given in Protocol 26. The influence of the various parameters on the security bounds and on the noise-robustness of our protocol is detailed in the next sections along with the constraints they must abide. Note that, similarly to the previous protocol, it is possible for the Client to choose the secret variables for all runs in advance and the other angles can be pre-computed. This is important when taking experimental implementations into account as it lowers the response delay of the Client and therefore increases the probability that the protocol terminates without the qubits decohering. Sampling these parameters in advance must take into account the probability of having to redo a given run j due to experimental defects. If p_{succ} is (an upper-bound on) the probability of not resetting a run on honest devices, then the Client needs to pre-sample $N = \mathcal{O}(n/p_{succ})$ runs with the same proportion of computation and test runs, d/n and t/n respectively.

We now show how this protocol can be extended to the case where the Server has more qubits in their quantum registers. This is a strict extension in the sense that the protocol presented above is a special case of the one from the next subsection on the specific two-qubit graph presented in Figure 7.1. The security properties of the general protocol will be directly applicable to the protocol constructed above.

7.2.4 Full Noise-Robust Verifiable Protocol

We start by showing how to extend the State Preparation step if the Client's and Server's communication registers can hold more than a single qubit and the optical link used to establish entanglement supports multiplexing. In that case, the state preparation consists of multiple parallel executions of the previously described state preparation protocols. Each qubit in one party's communication register is associated to one of the other party's qubits and they try to establish entanglement across each of those pairs of

[4] w would typically be set by the Client given its *a priori* understanding of the quality of the Server. As discussed further below, this does not affect security: a higher value will only induce more runs than necessary to achieve a given confidence level, while a lower value would risk aborting with high probability.

Protocol 26 VBQC for Small-Scale Quantum Internet

Client's Inputs:

- Two angles $(\phi(1), \phi(2))$ and a flow f on graph $G = (V = \{1,2\}; E = \{(1,2)\})$ describing the computation.
- A classical input to the computation $(x(1), x(2)) \in \{0,1\}^2$.

Protocol:

1. The Client chooses uniformly at random a partition (C, T) of $[n]$ $(C \cap T = \emptyset)$ with $\#C = d$, the sets of indices of the computation and test runs respectively.
2. For $j \in [n]$, the Client and the Server perform the following sub-protocol (if a party receives Redo_j from the other, both parties restart run j with fresh randomness):
   - If $j \in C$ (the run is a computation), they perform one instance of the Basic UBQC Protocol 25, let y_j be the classical output of this computation run (after corrections from measurement results). The Client may send message Redo_j to the Server before step 5 of this sub-protocol while the Server may send it to the Client at any time.
   - If $j \in T$ (the run is a test), the Client chooses $t_j \in_R \{1,2\}$ uniformly at random (the index of the trap qubit for this test run). Both parties then perform the following sub-protocol:
     a) The Client and the Server perform the same State Preparation steps as in the Basic UBQC Protocol 25, but with the Client applying the UBQC State Preparation for qubit t_j and the Dummy State Preparation for the other qubit. Let $(\tilde{\theta}(t_j), p(t_j))$ and p be their respective outputs, the Client sets $\theta(t_j) = \tilde{\theta}(t_j) + p(t_j)\pi + p\pi$ (if the dummy is in state $|1\rangle$ then the CZ operation will have the effect of exchanging $|+_\theta\rangle$ and $|-_\theta\rangle$).
     b) The Server performs a CZ gate between the qubits in both of its registers and sends Ack to the Client.
     c) For $v \in \{1,2\}$, the Client sends a measurement angle $\delta_j(v)$, the Server measures the appropriate register in the $\delta_j(v)$-basis, returning outcome $b_j(v)$ to the Client. The angle $\delta(v)$ is defined as follows: if $v \neq t_j$ (dummy qubit), $\delta_j(v) \in_R \Theta$ is chosen randomly; if $v = t_j$ (trap qubit), the Client samples $r_j(v) \in_R \{0,1\}$ and $\delta_j(v) = \theta_j(t_j) + r_j(v)\pi$.
     d) The Client sets the output of test run j as $s(t_j) := b(t_j) \oplus r(t_j)$.
3. For all $j \in T$ (test runs), the Client checks that $s(t_j) = 0$, incrementing a counter c_{fail} (initialised to 0) in case of failure. If $c_{fail} \geq w$, then the Client aborts, setting Abort as its output and sending it to the Server.
4. Otherwise, the Client checks whether there exists some value y such that $\#\{y_j \,|\, j \in C, y_j = y\} > d/2$. If such a value y exists, it sets it as its output and sends Ok to the Server. Otherwise it outputs Abort and sends it to the Server.

qubits. Since this process is heralded, if an entanglement establishment succeeded both parties know which qubits are entangled and can perform the next steps of the previous State Preparations (i.e. the Client measures its half of the EPR-pair either in a θ-basis for traps and computation qubits, or the computational basis for traps). The Server must then perform a SWAP gate (unless it is the final round of state preparation, in which case the qubits stay in the communication register) to transfer the state of the communication qubit to the correct position in its graph. This is then repeated until all qubits are initialised, with the Client reinitialising its qubits to the state $|+\rangle$ after each measurement.

This is described more precisely in the following Multi-Qubit State Preparation (Protocol 3). Without loss of generality we suppose that the size of the communication registers on the Client's and Server's sides are the same and can hold Q qubits, labelled $q^C_{C,l}$ on the Client's side and $q^S_{C,l}$ on the Server's side with $l \in [Q]$. We further make the assumption that the state of the qubits can be transferred from any position in the communication register of the Server to any position in the internal computation

register of the Server in an efficient way. We call $\mathsf{SWAP}(q, q')$ this operation between qubits q and q' and leave its implementation and optimisation to the Server as it is not necessarily done atomically. We also suppose that the qubits of the Server have all been initialised in the state $|0\rangle$ while those of the Client are in the state $|+\rangle$.

Algorithm 3 Multi-Qubit State Preparation Protocol

Inputs: The Client has as input a measurement basis $\mathbf{B}$ (either a random θ-basis or the computational basis).
Public Information: Number Q of qubits in communication registers, index q of destination qubit in Server's internal register.
Protocol:

1. The Client and the Server attempt to probabilistically establish entanglement (equivalent to CNOT) on any one of the pairs of qubits $(q^C_{C,l}, q^S_{C,l})$ for $l \in [Q]$. Let L be the index of the first established connection.
2. The Client performs the measurement described by basis $\mathbf{B}$ on qubit $q^C_{C,L}$ and records the outcome of measurement b_q. It sends Ack to the Server and reinitialises its qubit $q^C_{C,L}$ to the state $|+\rangle$.
3. If $q^S_{C,L} \neq q$, the Server performs the operation $\mathsf{SWAP}(q^S_{C,L}, q)$ and sends Ack to the Client. If $q^S_{C,L} = q$, no operation is performed. If q is a qubit in the communication register of the Server, it is treated by both Client and Server as part of the internal register from now on for future executions of the State Preparation Protocol.

We note that the Multi-Qubit State Preparation is a direct generalisation of the previous State Preparations that subsumes both the Dummy and UBQC State Preparation: if the size Q of the communication registers is equal to 1 we recover the previous protocols. Instead of reinitialising only the used qubit, the Client can also start anew at each run and reinitialise all its qubits as this may improve the fidelity of the generated EPR-pair. Any intermediate setting is also possible.

We now suppose that the Client has decided to perform a fixed computation which it has translated into a computational measurement pattern to be run on a graph G supported by the known topology of the Server's device. Given this graph G, we construct *test runs* based on a colouring of the graph. A partition of a graph in k sets – called colours – is a valid k-colouring if all adjacent vertices in the graph have different colours. We define it formally below.

Definition 7.1 (Graph Colouring)**.** *Let $G = (V, E)$ be a graph and for all $v \in V$ recall that $N_G(v)$ are the neighbours of v in G. Then a set of sets of vertices $\{V_i\}_{i\in[k]}$ with $V_i \subset V$ is a k-colouring if it is a partition of V which satisfies:*

$$\bigcup_{i=1}^{k} V_i = V \tag{7.1}$$
$$\forall i \in [k], \forall v \in V_i : N_G(v) \cap V_i = \emptyset$$

Hence, for each colour i, the Client can insert traps for all vertices of V_i while placing dummies in all other positions. This defines the test associated to colour i. It is easy to check that the traps inserted in this way are isolated from other qubits, thus giving deterministic outcomes when measured in their preparation basis, and that they are undetectable for the Server as a test run results for the Server in

applying the same sequence of operations as for the regular UBQC computation. The Client chooses a colouring $\{V_i\}_{i\in[k]}$ of G at the beginning of the protocol and sends it to the Server along with the graph G.

For each test run, after having chosen secretly choose a colour at random and sent rotated qubits for vertices of that colour and dummies everywhere else, the Client then instructs the Server to measure all qubits as in computation runs, but with the measurement angle of trap qubits corresponding to the basis they were prepared in and a random measurement basis for the dummies. A test run is now said to have passed if *all* the traps yield the expected measurement results, and is said to have failed otherwise. Figure 7.4 depicts one possible succession of computation and test runs for this new trap insertion technique.

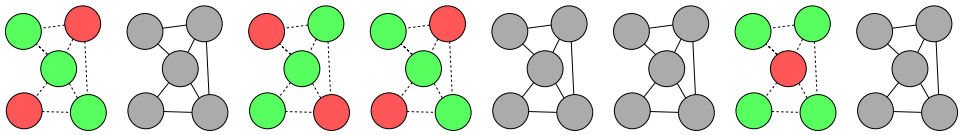


Figure 7.4: An example set of runs of the proposed protocol. Graphs in grey denote computation runs while graphs containing red nodes (traps) and green nodes (dummies) are test runs. This example graph on five nodes is completely covered with traps by the presented three types of test runs. Note that the Server remains completely oblivious of the differences between the runs, which are solely known to the Client.

We now describe the full protocol, replacing the State Preparation sub-protocol by direct qubit communication so as to not overload the description.

7.3 Security Results and Noise Robustness

We show that the protocol presented above is secure in the Abstract Cryptography Framework of [99], which guarantees automatically the full composability of protocols. As a consequence, the security of our protocol will hold in a wide range of situations of practical interest such as when different runs are distributed to different machines to reduce the overall execution time. This framework implies a higher standard of security than in other approaches (see e.g. [84] and Section 5.1 of [115]).

7.3.1 Security Analysis

The purpose of the final protocol described in this chapter is to emulate the Verifiable Blind Delegated Quantum Computation Resource while taking into account various degrees of limitations on the underlying hardware. We recall here the definition of this resource for the specific case of classical outputs.

Ideal Resource for Verifiable Delegated Quantum Computation. We define here a slightly modified version of Resource 12 presented in Section 3.3.3. The resource $\mathcal{S}$ has interfaces for two parties A and B. The A-interface takes two inputs: a classical input string x of length m and the description of $\mathcal{U}$, the unitary computation to perform on $n+k$ qubits (k of which are ancillae). The B-interface takes two input bits c and b, both of which are filtered and set to 0 in the honest case. If $c=0$, the output at A's interface is equal to (the classical result of) $\mathcal{M}_{Comp}\circ\mathcal{U}(|x\rangle|0\rangle^{\otimes k})$, where $\mathcal{M}_{Comp}$ denotes

Protocol 27 Noise-Robust VBDQC with Deterministic Classical Output

Client's Inputs:

- Angles $\{\phi(v)\}_{v\in V}$ and flow f on graph $G=(V,E,I,O)$.
- Classical input to the computation $x\in\{0,1\}^{\#I}$.

Protocol:

1. The Client chooses uniformly at random a partition (C,T) of $[n]$ ($C\cap T=\emptyset$) with $\#C=d$, the sets of indices of the computation and test runs respectively.
2. For $j\in[n]$, the Client and the Server perform the following sub-protocol (the Client may send message Redo_j to the Server before step 2.c while the Server may send it to the Client at any time, both parties then restart run j with fresh randomness):
   (a) If $j\in T$ (test), the Client chooses uniformly at random a colour $\mathsf{V}_j\in_R\{V_k\}_{k\in[K]}$ (this is the set of traps for this test run).
   (b) The Client sends $\#V$ qubits to the Server. If $j\in T$ and the destination qubit $v\notin\mathsf{V}_j$ is a non-trap qubit (therefore a dummy), then the Client chooses uniformly at random $d_j(v)\in_R\{0,1\}$ and sends the state $|d_j(v)\rangle$. Otherwise, the Client chooses at random $\theta_j(v)\in_R\Theta$ and sends the state $\left|+_{\theta_j(v)}\right\rangle$.
   (c) The Server performs a CZ gate between all its qubits corresponding to an edge in the set E.
   (d) For $v\in V$, the Client sends a measurement angle $\delta_j(v)$, the Server measures the appropriate corresponding qubit in the $\delta_j(v)$-basis, returning outcome $b_j(v)$ to the Client. The angle $\delta_j(v)$ is defined as follows:
   - If $j\in C$ (computation), it is the same as in UBQC, computed using the flow and the computation angles $\{\phi(v)\}_{v\in V}$. For $v\in I$ (input qubit) the Client uses $\tilde{\theta}_j(v)=\theta_j(v)+x(v)\pi$ in the computation of $\delta_j(v)$.
   - If $j\in T$ (test): if $v\notin\mathsf{V}_j$ (dummy qubit), it is chosen uniformly at random from Θ; if $v\in\mathsf{V}_j$ (trap qubit), the Client chooses uniformly at random $r_j(v)\in_R\{0,1\}$ and sets $\delta_j(v)=\theta_j(v)+r_j(v)\pi$.
3. For all $j\in T$ (test run) and $v\in\mathsf{V}_j$ (traps), the Client verifies that $b_j(v)=r_j(v)\oplus d_j(v)$, where $d_j(v)$ is defined for non-dummy qubits by $d_j(v)=\bigoplus_{\tilde{v}\in N_G(v)}d_j(\tilde{v})$ is the sum over the values of neighbouring dummies of qubit v. Let c_{fail} be the number of failed test runs (where at least one trap qubit does not satisfy the relation above), if $c_{fail}\geq w$ then the Client aborts by sending message Abort to the Server.
4. Otherwise, let $y_j\in\{0,1\}^{\#O}$ for $j\in C$ be the classical output of computation run j (after corrections from measurement results). The Client checks whether there exists some output value y such that $\#\{y_j\,|\,j\in C,\,y_j=y\}>d/2$. If such a value y exists (this is then the majority output), it sets it as its output and sends message Ok to the Server. Otherwise it sends message Abort to the Server.

a computational basis measurement. When $c=1$, A receives the Abort message. At B's interface, $\mathcal{S}$ outputs nothing for $b=0$ while for $b=1$, B receives $l(\mathcal{U},x)$, the permitted leakage.

The permitted leakage in the case of our protocol is set to G, the graph used in the computation, and the size of the input m. When G is a universal graph for MBQC computation, the permitted leakage reduces to an upper-bound on the size of the computation $\mathcal{U}$.

Instead of following the direct approach to proving security (describing a simulator for malicious parties), we will take a slightly different path. Proving security will rely on the results presented in [41] and recalled in Section 3.4, which reduces the security of a Delegated Quantum Computation Protocol to the conjunction of four stand-alone criteria:

- ϵ_{cor}-local-correctness, which is satisfied if the protocol with honest players outputs the expected

output;

- ϵ_{bl}-local-blindness, meaning that the malicious Server's state at the end of the protocol is indistinguishable from the one which it could have generated on its own;
- ϵ_{ver}-local-verifiability, if either the Client accepts a correct computation or aborts at the end of the protocol.
- ϵ_{ind}-independent-verification, i.e. the Server can determine on its own, using the transcript of the protocol and its internal registers, whether the Client will decide to abort or not.

With this at hand, we now state our main result:

Theorem 7.1 (Security of Protocol 27)**.** *For $n = d + t$ such that d/n and t/n are fixed in $(0, 1)$ and w such that w/t is fixed in $(0, 1/2k)$, Protocol 27 with d computation runs, t test runs, and a maximum number of tolerated failed test runs of w is ϵ-composably-secure with $\epsilon = 4\sqrt{2\epsilon_{ver}}$ and with ϵ_{ver} exponentially small in n.*

Proof. We show that our protocol satisfies each of the stand-alone criteria before combining them to get composable security.

Perfect Local-Correctness. On perfect (non-noisy) devices, local-correctness is implied by the correctness of the underlying UBQC Protocol. This is because all the completed computation runs correspond to the same deterministic UBQC computation, and that on such devices, general UBQC Protocols have been proven to be perfectly correct [21, 41]. Thus $\epsilon_{cor} = 0$.

Perfect Local-Blindness. To prove that Equation 3.9 holds for $\epsilon_{bl} = 0$, first note that at the end of our protocol, the Client A reveals to the Server B whether the computation was accepted or aborted. Hence, each case can be analysed separately. Second, we show that the interrupted runs that have triggered a Redo can be safely ignored. Indeed, each one of them is the beginning of an interrupted UBQC computation, and, because UBQC is composable and perfectly blind [41], no information can leak to the Server through the transmitted qubits. In addition, our protocol restricts the honest party A in its ability to emit Redo requests, so that no correlations are created between the index of the interrupted runs and $\mathcal{U}$ or the secret random parameters used in the runs (angle and measurement padding, and trap preparations). As a consequence, from the point of view of B, the state of the interrupted runs is completely independent of the state of the non-interrupted ones and does not contain information regarding the input, computation or secret parameters. That is, its partial trace over A can be generated by B alone.

For the non-interrupted runs, we can invoke the same kind independence argument between the computation runs and the test runs. As a result blindness of our protocol stems from the blindness of the underlying computation runs. In case the full protocol is a success, we can rely on the composability of the perfect blindness of each UBQC computation run to have perfect local-blindness. For an abort, we can consider a situation that is more advantageous for B by supposing that alongside the Abort message sent by A, it also gives away the location of the trap qubits. In this modified situation, the knowledge of the computation being aborted does not bring additional information to B as it only reveals that one of the attacked position was a trap qubit, which B now already knows. Using our independence argument between trap location on the one hand and the inputs, computation and other

secret parameters, we conclude that revealing the location of the trap qubits does not affect the blindness of the computation runs. Hence, using composability again and combining the abort and accept cases, we arrive at Equation 3.9 with $\epsilon_{bl} = 0$.

Perfect Local-Independent-Verification. Because in our protocol, the Client shares with the Server whether the computation was a success or an abort, this is trivially verified.

Exponential Local-Verifiability (Proof Sketch). Local-verifiability is satisfied if any deviation by the possibly malicious Server yields a state that is ϵ_{ver}-close to a mixture of the correct output and the Abort message. Equivalently, the probability that the Server makes the Client accept an incorrect outcome is bounded by ϵ_{ver}. Let d/n, t/n and w/t be the ratios of test, computation and tolerated failed test runs. Our protocol's local-verifiability is given by Lemma 7.1. We give below a sketch of the main ideas yielding the result.

The first step is to describe all the messages received by the Client during the execution of the protocol without making assumptions on the behaviour of the Server. This comprises the outcomes of the computational runs, but also the measurement of trap qubits and of any other qubit used in the computation or in the tests. Following [53], this can be expressed as the state one would obtain in the perfect protocol followed by a pure deviation on this state.

The second step consists of using this state to bound the probability of failure, i.e. the probability of accepting the computation but having the wrong result. This happens if at least $d/2$ outcomes of the computation runs have had at least one bit-flip, and no more than w test runs have failed.

In the third step, we use the randomisation over the prepared qubits and measurement angles to twirl the deviation of the Server and to reduce it to diagonal form in the Pauli basis. This further simplifies the expression for the bound.

The fourth step exploits this reduced form of the attack by noticing that it can be dealt with in a classical fashion. To this end, possible attacks are classified using two criteria:

1. Does the attack affect at least $d/2$ computation runs? Only such attacks stand a chance to corrupt the result of the computation, otherwise the repetition code automatically corrects the deviations.
2. Does the attack make less than w of the t test runs fail? Only then will the deviations be tolerated without triggering a client-side abort.

Depending on the answer to the questions above, the attack falls in one of four regimes. Optimally, the protocol would abort if and only if the result of the computation is corrupted. Clearly, we cannot hope to perfectly achieve this. We therefore must take into account two types of incorrect categorisation. A *false positive* happens when the protocol aborts although less than $d/2$ computation runs have been affected. While this is undesirable behaviour in terms of noise-tolerance, it does not affect security. Since we are here analysing the verifiability of our protocol, we are solely concerned about *false negatives*: the attack affects at least $d/2$ computation runs and no abort is triggered. To achieve a satisfying level of security, no attack should fall into this regime with more than negligible probability.

For intuition's sake, we give here an analysis of the average case (Figure 7.5). If the Server deviates in exactly half of the runs on a single qubit (which we suppose to be sufficient for corrupting the computation), the number of affected computation runs will be $d/2$ on average. In other words, there are

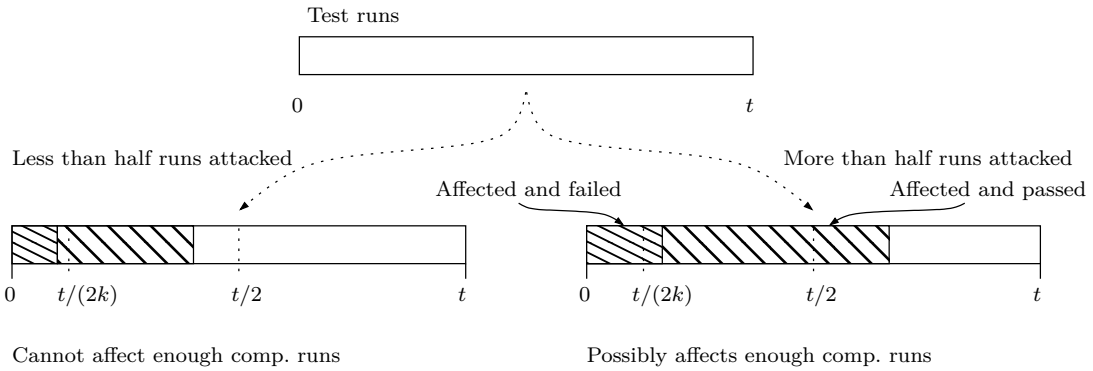


Figure 7.5: Because of blindness, an attack on less than half of the runs is likely to affect less than $d/2$ computation runs. As such attacks are error-corrected, the protocol should output a result. On the contrary, if an attack is performed on more than half the runs, it has a chance to corrupt the computation and the result should not be trusted. Likewise, in the first case, less than $t/2$ test runs should be affected, while in the second more than $t/2$ should be. Yet, there are only $1/k$ test qubits per test run meaning that affected test runs turn into failed test runs with $1/k$ probability. If the threshold value w is set below $t/2k$, aborts are thrown only when there is a effective risk of tampering. Here, we arbitrarily set $k = 4$.

good chances that just enough computation runs are affected to corrupt the final result. This is because the attacker is blind and hence the deviations are randomly distributed over computation and test runs. Similarly, we expect the number of affected test runs to be $t/2$. Considering that any qubit in a run has a probability of $1/k$ to be a trap, we expect the number of failed test runs, i.e. the number of test runs with at least one affected trap, to be $t/2k$. As a consequence, choosing $w \geq t/2k$ cannot lead to a secure protocol, since the simple attack described above has a non-negligible probability of corrupting the final result while remaining unnoticed. Conversely, setting $w \leq t/2k$ foils this strategy.

The proof presented below goes beyond this average case analysis by showing that this attack is essentially optimal. It uses concentration bounds for the underlying probability distributions to obtain precise bounds that are exponentially-low in the various parameters.

Proof of Exponential Composable-Security. Our protocol has perfect correctness (for noiseless devices), blindness and input-independent verification. In addition, it is ϵ_{ver}-locally-verifiable with ϵ_{ver} exponentially small in n. Therefore, by the Local-Reduction Theorem, it is ϵ-composably-secure with $\epsilon = \delta = 4\sqrt{2\epsilon_{ver}}$ and ϵ exponentially small in n.

■

Note that because we used the Local-Reduction Theorem to obtain fully composable security, we incur an additional square root on our verifiability bound given by Equation 3.15 and need to satisfy the additional independence property. This is of course not required if the protocol is only used sequentially with other schemes, which will probably be the case in early quantum computations since the machines will not be able to handle multiple protocols at the same time. In this case, the stand-alone model would be sufficient since it provides sequential composition, but would fail if parallel composition is needed.

We now prove formally the verifiability of our protocol.

Lemma 7.1 (Local Verifiability of Protocol 27)**.** *Let $0 < w/t < 1/2k$ and $0 < d/n < 1$ be fixed ratios, for a k the number of different test runs. Protocol 27 is ϵ_{ver}-locally-verifiable for exponentially-low ϵ_{ver}.*

Proof. We will take a direct approach for proving Lemma 7.1 by bounding the probability of yielding a wrong output while not aborting. To do so, we consider the state of the *combined computation* as if it were a single verified computation and not made of separate sequential runs. Once again, in our protocol, because the parties can only ask for redoing a run independently of the input, computation, used randomness and of the output of the computation itself (comprising the result of trap measurements), interrupted runs can be safely ignored in the verification analysis as the state corresponding to these runs is uncorrelated to that of the completed runs. The combined computation view will be useful as we want to consider the Server B performing any kind of attack. In particular, it could decide to perform some action on a qubit given measurements in one or several of the underlying runs, or to entangle the various underlying runs together. Yet, for each qubit of the combined computation, we will continue to refer to the underlying run this qubit would belong to if the computation was done using sequential runs.

Output of the combined computation. We first consider the density operator $B(\{\mathcal{F}_j\}_j, \nu)$ that corresponds to the classical messages the Client A receives during its interaction with B, comprising the final message containing the encrypted measurement outcomes. Below, the CPTP maps $\{\mathcal{F}_j\}_j$ represent the chosen behavior of B and possibly act on the combined computation as a whole, and not only run by run. By representing the classical messages as quantum states in the computational basis, we can always write:

$$B(\{\mathcal{F}_j\}_j, \nu) = \mathrm{Tr}_B \left\{ \sum_b |b + c_r\rangle\langle b| \, \mathcal{F}\mathcal{P} \left(|0\rangle\langle 0|_B \otimes \left|\Psi^{\nu,b}\right\rangle\left\langle\Psi^{\nu,b}\right| \right) \mathcal{P}^\dagger \mathcal{F}^\dagger \, |b\rangle\langle b + c_r| \right\} \tag{7.2}$$

where b is the list of measurement outcomes defining the computation branch; ν is a composite index relative to the secret parameters chosen by A, i.e. the type of each underlying run, the padding of the measurement angles and measurements outcomes and the trap setup; $|b + c_r\rangle\langle b|$ ensures that only the part corresponding to the current computation branch is taken into account and removes the One-Time-Pad encryption on non-output and non-trap qubits while leaving output and trap qubits unaffected, i.e. encrypted; $|0\rangle\langle 0|_B$ is some internal register for B in a fixed initial state; and $\left|\Psi^{\nu,b}\right\rangle$ is the state of the qubits sent by A to B at the beginning of the protocol tensored with quantum states representing the measurement angles of the computation branch b.

To obtain this result, we can follow the line of proof of [53] and [78] applied to the combined computation. This works by noting that for a given computation branch b and given random parameters ν, all the measurement angles are fully determined. Therefore, provided that the computation branch is b, we can include the measurement angles into the initial state. This defines $\left|\Psi^{\nu,b}\right\rangle$. Then, each $\mathcal{F}_j$ is decomposed into an honest part and a pure deviation. All the deviations are commuted and collected into $\mathcal{F}$ applied after $\mathcal{P}$, the unitary part of honest protocol, is applied. The projections onto $|b\rangle$ then ensures that after the deviation induced by B the perceived computation branch is b. This, together with the decrypting of non-output non-trap qubits, gives Equation 7.2.

Probability of failure. A failure for the combined computation occurs when the result after the majority vote is incorrect while the computation is accepted.

The combined computation being deterministic, we can define $P_\perp$, the projector onto the subspace of incorrect states for the output qubits before the majority vote. Yet, for the combined computation to be accepted, no more than w test runs have a trap qubit measurement outcome opposite to what was expected. Let $\mathtt{T}$ denote the set of trap qubits which is determined by T, the set of test runs, and the type of each test run. In absence of any deviation on the combined computation, their expected value is $|r_\mathtt{T}\rangle = \bigotimes_{\mathtt{t}\in\mathtt{T}} |r_\mathtt{t}\rangle$ where $r_\mathtt{T} = (r_\mathtt{t})_{\mathtt{t}\in\mathtt{T}}$ denotes the measurement outcome padding values restricted to trap qubits. Therefore, the projector onto the states of the trap qubits yielding to an accepted combined computation can be written as $\sum_{\mathtt{w}\in\mathtt{W}} X_\mathtt{T}^\mathtt{w} |r_\mathtt{T}\rangle\langle r_\mathtt{T}| X_\mathtt{T}^\mathtt{w}$ with $X_\mathtt{T}^\mathtt{w} = \bigotimes_{\mathtt{t}\in\mathtt{T}} X_\mathtt{t}^{\mathtt{w}_\mathtt{t}}$, and where $\mathtt{W}$ is the set of length $\#\mathtt{T}$ binary vectors $\mathtt{w}$ such that $\mathrm{w_H}(\mathtt{w}) < w$, i.e. that have a one in no more than w underlying (test) runs. Combining the projector onto incorrect output and the one for accepted computation, we obtain the probability of failure:

$$\begin{aligned}\Pr(\text{fail}) = \sum_\nu \sum_{\mathtt{w}\in\mathtt{W}} \sum_{b,k,\sigma,\sigma'} \Pr(\nu) \operatorname{Tr}\Big\{ (P_\perp \otimes X_\mathtt{T}^\mathtt{w} |r_\mathtt{T}\rangle\langle r_\mathtt{T}| X_\mathtt{T}^\mathtt{w}) \cdot \\ \left(\alpha_{k\sigma}\alpha_{k\sigma'}^* |b + c_r\rangle\langle b| \sigma\mathcal{P} \left|\Psi^{\nu,b}\right\rangle\!\left\langle\Psi^{\nu,b}\right| \mathcal{P}^\dagger\sigma' |b\rangle\langle b + c_r|\right) \Big\}\end{aligned} \tag{7.3}$$

where $\mathcal{F}$ has been decomposed into Kraus operators indexed by k, that were in turn decomposed onto the Pauli basis through the coefficients $\alpha_{k\sigma}$ and $\alpha_{k\sigma'}$. Consequently, σ and σ' are Pauli matrices.

Necessary condition for failure. The difficulty with the above expression for the probability of failure consists in determining the exact form of $P_\perp$ and manipulating it. Instead, we will derive a coarse necessary condition for the final state of the non-trap qubits to be in the subspace defined by $P_\perp$. Then, we will upper bound Pr(fail) by evaluating the probability of satisfying our necessary condition while accepting the whole computation.

First, note that the output of the computation being classical and deterministic, we can write the correct decrypted output state as $|s_\mathtt{O}\rangle\langle s_\mathtt{O}|$ for some length $|\mathtt{O}|$ binary vector $s_\mathtt{O}$ over the set of output qubit positions $\mathtt{O}$ of the combined computation. Next, as for the trap qubits, the value sent to the Client is One-Time-Padded by the value of the random parameter $r_\mathtt{O}$ to preserve blindness of the Server (i.e. c_r is 0 for output qubits). Hence, the state of the output qubits received by the Client in absence of deviation is $|s_\mathtt{O} + r_\mathtt{O}\rangle\langle s_\mathtt{O} + r_\mathtt{O}|$.

Now, because the result of the computation is the majority vote of the measurement outcomes for the output qubit for each underlying computation run, each result bit is protected by a length d repetition code. All attacks resulting in less than $d/2$ non-trivially affected underlying computational runs will be corrected. Conversely, for a failure to happen, it is necessary that at least $d/2$ underlying computation runs are non-trivially affected by the attack Ω. This means that the subspace stabilized by $P_\perp$ is also stabilized by the coarser projection operator $\sum_{\mathtt{v}\in\mathtt{V}} X_\mathtt{O}^\mathtt{v} |s_\mathtt{O} + r_\mathtt{O}\rangle\langle s_\mathtt{O} + r_\mathtt{O}| X_\mathtt{O}^\mathtt{v}$, where $\mathtt{V}$ is the set of binary vectors $\mathtt{v}$ over $\mathtt{O}$ with $\mathrm{w_H}(\mathtt{v}) \geq d/2$, i.e. with ones in at least $d/2$ positions of computational runs. As a

consequence, the latter projector can be used to replace $P_\perp$ which yields an upper bound on Pr(fail):

$$\begin{aligned}\Pr(\text{fail}) \leq \sum_{\substack{\nu \\ \mathtt{v}\in\mathtt{V},\mathtt{w}\in\mathtt{W} \\ b',k,\sigma,\sigma'}} \Pr(\nu)\alpha_{k\sigma}\alpha^*_{k\sigma'} \operatorname{Tr}\Big\{ & (X_{\mathtt{0}}^{\mathtt{v}} \otimes X_{\mathtt{T}}^{\mathtt{w}}) \left(|s_{\mathtt{0}} + r_{\mathtt{0}}\rangle\langle s_{\mathtt{0}} + r_{\mathtt{0}}| \otimes |r_{\mathtt{T}}\rangle\langle r_{\mathtt{T}}|\right) (X_{\mathtt{0}}^{\mathtt{v}} \otimes X_{\mathtt{T}}^{\mathtt{w}}) \cdot \\ & |b + c_r\rangle\langle b|\, \sigma\mathcal{P} \left|\Psi^{\nu,b}\right\rangle\!\left\langle\Psi^{\nu,b}\right| \mathcal{P}^\dagger \sigma' \, |b\rangle\langle b + c_r| \Big\}\end{aligned} \tag{7.4}$$

where b' is the binary vector obtained from b by restricting it to non-output, non-trap qubits, i.e. $b = (b_O, b_T, b')$. Since $c_r = 0$ for output and trap qubits, the above equation is obtained from the simple equality $\sum_b (\langle s_{\mathtt{0}} + r_{\mathtt{0}}| \otimes \langle r_{\mathtt{T}}|)(X_{\mathtt{0}}^{\mathtt{v}} \otimes X_{\mathtt{T}}^{\mathtt{w}}) |b + c_r\rangle\langle b| = \sum_{b'} (|s_{\mathtt{0}} + r_{\mathtt{0}}\rangle \otimes |r_{\mathtt{T}}\rangle \otimes |b'\rangle)(X_{\mathtt{0}}^{\mathtt{v}} \otimes X_{\mathtt{T}}^{\mathtt{w}})$ and the circularity of the trace.

Using blindness of the scheme. The design of the protocol yielding the combined computation ensures blindness. This implies that the resulting state of any set of qubits after applying $\mathcal{P}$ and taking the average over their possible random preparations parameters is a completely mixed state. This can be applied in the above equation for the set of non-output and non-trap qubits. For output and trap qubits, we need first to compute inner products before taking the sum over their random preparation parameters $\nu_{\mathtt{0}}$ and $\nu_{\mathtt{T}}$ respectively. However, we know that the perfect protocol produces the traps in their expected states $|s_{\mathtt{o}} + r_{\mathtt{o}}\rangle$ and $|r_{\mathtt{t}}\rangle$. This gives, using the circularity of the trace:

$$\begin{aligned}\Pr(\text{fail}) \leq \sum_{\substack{\nu_{\mathtt{0}},\nu_{\mathtt{T}} \\ \mathtt{v}\in\mathtt{V},\mathtt{w}\in\mathtt{W} \\ b',k,\sigma,\sigma'}} \Pr(\nu_{\mathtt{0}},\nu_{\mathtt{T}})\alpha_{k\sigma}\alpha^*_{k\sigma'} \Big[& \langle s_{\mathtt{0}} + r_{\mathtt{0}}| \otimes \langle r_{\mathtt{T}}| \otimes \langle b'| \, (X_{\mathtt{0}}^{\mathtt{v}} \otimes X_{\mathtt{T}}^{\mathtt{w}}) \\ & \left[\sigma \left(|s_{\mathtt{0}} + r_{\mathtt{0}}\rangle\langle s_{\mathtt{0}} + r_{\mathtt{0}}| \otimes |r_{\mathtt{T}}\rangle\langle r_{\mathtt{T}}| \otimes \frac{\mathsf{I}}{\operatorname{Tr}\mathsf{I}}\right) \sigma'\right] (X_{\mathtt{0}}^{\mathtt{v}} \otimes X_{\mathtt{T}}^{\mathtt{w}}) \, |s_{\mathtt{0}} + r_{\mathtt{0}}\rangle \otimes |r_{\mathtt{T}}\rangle \otimes |b'\rangle \Big]\end{aligned} \tag{7.5}$$

As the Pauli matrices are traceless, this imposes $\sigma_l = \sigma'_l$ for $l \notin \mathtt{0} \cup \mathtt{T}$, where subscript l is used to select the action of σ and σ' on qubit l. For output qubits, $\sum_{r_{\mathtt{o}}} \langle s_{\mathtt{o}} + r_{\mathtt{o}}| \, X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \sigma_{\mathtt{o}} \, |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle\langle s_{\mathtt{o}} + r_{\mathtt{o}}| \, \sigma'_{\mathtt{o}} X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \, |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle$ vanishes for $\sigma_{\mathtt{o}} \neq \sigma'_{\mathtt{o}}$ and similarly for the traps, $\sum_{r_{\mathtt{t}}} \langle r_{\mathtt{t}}| \, X_{\mathtt{t}}^{\mathtt{w}_{\mathtt{t}}} \sigma_{\mathtt{t}} \, |r_{\mathtt{t}}\rangle\langle r_{\mathtt{t}}| \, \sigma'_{\mathtt{t}} X_{\mathtt{t}}^{\mathtt{w}_{\mathtt{t}}} \, |r_{\mathtt{t}}\rangle$ vanishes for $\sigma_{\mathtt{t}} \neq \sigma'_{\mathtt{t}}$. Hence we get:

$$\begin{aligned}\Pr(\text{fail}) \leq & \sum_{\substack{\nu_{\mathtt{0}},\nu_{\mathtt{T}} \\ \mathtt{v}\in\mathtt{V},\mathtt{w}\in\mathtt{W} \\ k,\sigma}} \Pr(\nu_{\mathtt{0}},\nu_{\mathtt{T}}) |\alpha_{k\sigma}|^2 \times \prod_{\mathtt{o}\in\mathtt{0}} |\langle s_{\mathtt{o}} + r_{\mathtt{o}}| X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \sigma_{\mathtt{o}} |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle|^2 \times \prod_{\mathtt{t}\in\mathtt{T}} |\langle r_{\mathtt{t}}| X_{\mathtt{t}}^{\mathtt{w}_{\mathtt{t}}} \sigma_{\mathtt{t}} |r_{\mathtt{t}}\rangle|^2 \\ \leq & \sum_{k,\sigma} |\alpha_{k\sigma}|^2 f(\sigma)\end{aligned} \tag{7.6}$$

with

$$f(\sigma) = \sum_{\substack{\nu_{\mathtt{0}},\nu_{\mathtt{T}} \\ \mathtt{v}\in\mathtt{V},\mathtt{w}\in\mathtt{W}}} \Pr(\nu_{\mathtt{0}},\nu_{\mathtt{T}}) \times \prod_{\mathtt{o}\in\mathtt{0}} |\langle s_{\mathtt{o}} + r_{\mathtt{o}}| X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \sigma_{\mathtt{o}} |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle|^2 \times \prod_{\mathtt{t}\in\mathtt{T}} |\langle r_{\mathtt{t}}| X_{\mathtt{t}}^{\mathtt{w}_{\mathtt{t}}} \sigma_{\mathtt{t}} |r_{\mathtt{t}}\rangle|^2 \tag{7.7}$$

Worst case scenario (for the upper-bound). The worst case scenario corresponds to maximising the bound on Pr(fail). Since we have $\sum_{k,\sigma} |\alpha_{k\sigma}|^2 = 1$, our bound is worst when $\alpha_{k\sigma} = 1$ for σ such that the value of $f(\sigma)$ is maximum. In this case we get:

$$\Pr(\text{fail}) \leq \max_{\sigma} f(\sigma) \tag{7.8}$$

Simplified expression for the bound. Given our protocol, a global trap and output qubit configuration $\nu_{\mathtt{O}}, \nu_{\mathtt{T}}$ is defined by (i) the set $\mathtt{T}$ of trap qubits, itself entirely determined by the position and kind of test runs within the sequence of runs, and (ii) the preparation parameters θ_l and r_l of each trap and output qubits. Each parameter of (i) and (ii) being chosen independently, the probability of a given configuration $\nu_{\mathtt{O}}, \nu_{\mathtt{T}}$ can be decomposed into the probability $\Pr(\mathtt{T})$ for a given configuration of trap locations multiplied by the probability of a given configuration for the prepared state of the trap and output qubits, $\prod_{l\in\mathtt{O}\cup\mathtt{T}} \sum_{\theta_l,r_l} \Pr(\theta_l, r_l)$. Using this, one can rewrite $f(\sigma)$:

$$f(\sigma) = \sum_{\substack{\mathtt{T}\\ \mathtt{v}\in\mathtt{V},\mathtt{w}\in\mathtt{W}}} \Pr(\mathtt{T}) \times \prod_{\mathtt{o}\in\mathtt{O}} \sum_{\theta_{\mathtt{o}},r_{\mathtt{o}}} \Pr(\theta_{\mathtt{o}}, r_{\mathtt{o}}) |\langle s_{\mathtt{o}} + r_{\mathtt{o}}| X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \sigma_{\mathtt{o}} |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle|^2 \times \prod_{\mathtt{t}\in\mathtt{T}} \sum_{\theta_{\mathtt{t}},r_{\mathtt{t}}} \Pr(\theta_{\mathtt{t}}, r_{\mathtt{t}}) |\langle r_{\mathtt{t}}| X_{\mathtt{t}}^{\mathtt{w}_{\mathtt{t}}} \sigma_{\mathtt{t}} |r_{\mathtt{t}}\rangle|^2 \tag{7.9}$$

Now, let σ be a maximising attack and denote by $\sigma_{|X}$ the binary vector indexed by qubit positions of the combined computation where ones mark qubit positions for which σ acts as X or Y. In the following, we allow $\mathtt{O}$ to also denote the binary vector over qubit positions of the combined computation where ones are positioned for qubits in O, and similarly for $\mathtt{T}$. Using the fact that $|\langle s_{\mathtt{o}} + r_{\mathtt{o}}| X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \sigma_{\mathtt{o}} |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle|^2$ is equal to 1 for $X_{\mathtt{o}}^{v_{\mathtt{o}}} \sigma_{\mathtt{o}} \in \{I, Z\}$ and 0 otherwise, we obtain that

$$\prod_{\mathtt{o}\in\mathtt{O}} \sum_{\theta_{\mathtt{o}},r_{\mathtt{o}}} \Pr(\theta_{\mathtt{o}}, r_{\mathtt{o}}) |\langle s_{\mathtt{o}} + r_{\mathtt{o}}| X_{\mathtt{o}}^{\mathtt{v}_{\mathtt{o}}} \sigma_{\mathtt{o}} |s_{\mathtt{o}} + r_{\mathtt{o}}\rangle|^2 = \begin{cases} 1 & \text{for } \mathtt{O} \odot \sigma_{|X} = \mathtt{v} \\ 0 & \text{otherwise} \end{cases} \tag{7.10}$$

where, for a and b binary vectors, $a \odot b$ is the bit-wise binary product vector. We obtain a similar expression for the trap qubits. Inserting these expressions in Equation 7.7 we obtain that for the attack to be successful, it must affect in a non-trivial way at least $d/2$ computation runs, and at most w test runs. The probability of failure can thus be rewritten as:

$$\Pr(\text{fail}) \leq \max_{\sigma} \sum_{\mathtt{T}\in\Upsilon_\sigma} \Pr(\mathtt{T}) \tag{7.11}$$

where Υ_σ are configurations where the binary vector $\sigma_{|X}$ has ones in at most w test runs and has ones on at least $d/2$ computation runs.

Closed form upper bound. Now, assume that the maximum above is attained for some σ that happen to affect one of the run, say k, on more than one qubit. Consider σ' with the sole difference to σ that only one of the qubits in run k is affected by the attack. Because run k is still non trivially affected by σ and σ', we conclude that all configurations $\mathtt{T}$ in Υ_σ are also in $\Upsilon_{\sigma'}$. Therefore

$$\Pr(\text{fail}) \leq \max_{m} \max_{\sigma\in E_m} \sum_{T\in\Upsilon_\sigma} \Pr(T) \tag{7.12}$$

where E_m denotes the set of Pauli operators with m single qubit deviations all in distinct runs. Note that the parameter m and the locations of the attacks within each run describe the adversary's strategy.

Additionally, since the random choice of test runs is completely uniform, the term $\sum_{T\in\Upsilon_\sigma}\Pr(T)$ is invariant under permutations of the test and computation runs. We can hence restrict the range of the maximum to the specific Pauli operators σ_m with a deviation on a single qubit in each of the first m runs:

$$\Pr(\text{fail}) \leq \max_m \sum_{T\in\Upsilon_{\sigma_m}} \Pr(T) \tag{7.13}$$

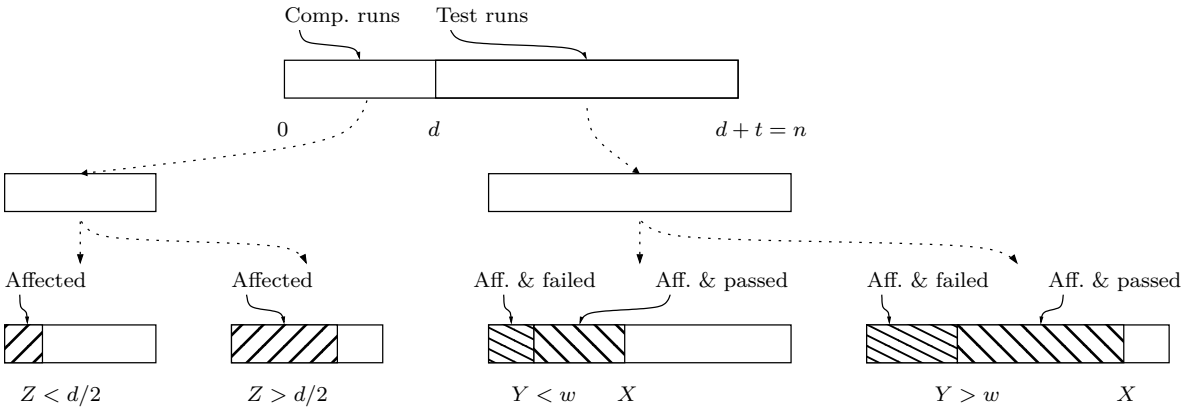


Figure 7.6: The four cases needed to determine a closed form upper bound for the probability of failure. First, we determine the probability for the number of affected computation runs. If it is low enough ($Z < d/2$), no need to abort. If it is high ($Z \geq d/2$), we find a bound on the probability that the number of failed test runs Y is below or above w.

Formal bound. To find a closed form upper bound for the right-hand side of Equation 7.13, we fix an arbitrary threshold $(1/2-\phi)\,n$ for m controlled by the (small) parameter $\phi > 0$ and distinguish two regimes (see Figure 7.6):

1. For $m \leq (1/2-\phi)\,n$, we use an upper bound on the probability that σ_m manages to affect at least $d/2$ computation runs, then ignoring the condition that σ_m affects less than w traps.
2. For $m \geq (1/2-\phi)\,n$, we conversely upper-bound the probability that σ_m triggers not more than w traps, thus ignoring the condition that σ_m affects more than $d/2$ computation runs. This is done in two steps:

   a) Find a lower bound (with high probability) on the number of affected test runs.
   b) Use this number to find a lower bound on the number of triggered traps.

In the following, we define the constant ratios of test, computation and tolerated failed test runs as $\delta := d/n$, $\tau := t/n$ and $\omega := w/t$. Let Z be a random variable counting the number affected computation runs and Y a random variable counting the number of failed test runs, i.e. the number of affected test

runs where the deviation hits a trap. We have that:

(7.14)
$$\begin{aligned}\Pr[\text{fail}] \leq \max_m \sum_{T\in\Upsilon_{\sigma_m}} \Pr(T) &= \max_m \Pr\left[Z \geq \frac{d}{2} \wedge Y \leq w\right] \\ &= \max\left\{\max_{m\leq\left(\frac{1}{2}-\phi\right)n} \Pr\left[Z \geq \frac{d}{2} \wedge Y \leq w\right], \max_{m\geq\left(\frac{1}{2}-\phi\right)n} \Pr\left[Z \geq \frac{d}{2} \wedge Y \leq w\right]\right\} \\ &\leq \max\left\{\max_{m\leq\left(\frac{1}{2}-\phi\right)n} \Pr\left[Z \geq \frac{d}{2}\right], \max_{m\geq\left(\frac{1}{2}-\phi\right)n} \Pr[Y \leq w]\right\}\end{aligned}$$

Since $\Pr[Z \geq d/2]$ and $\Pr[Y \leq w]$ are respectively increasing and decreasing with the number of attacked runs, both inner maximums are attained for $m = (1/2 - \phi)\, n$ and we therefore focus on this case.

We start by computing a bound on the probability too many of the computation runs failed. Note that Z is (n, d, m)-hypergeometrically distributed (of mean md/n). We are then interested in the following probability which we can upper-bounded using Corollary 2.2 (Serfling's bound) for $m/n < 1/2$:

(7.15)
$$\Pr\left[Z \geq \frac{d}{2}\right] \leq \exp\left(-2\left(\frac{1}{2} - \frac{m}{n}\right)^2 \frac{d^2}{m}\right) \leq \exp\left(-\frac{4\delta^2\phi^2}{1-2\phi}n\right)$$

We then focus on upper-bounding the probability that not enough test runs failed. We start by defining the random variable X counting the test runs affected by the Server's deviation. Then, for any threshold x we have that:

(7.16)
$$\Pr[Y \leq w] = \Pr[Y \leq w \mid X < x] \Pr[X < x] + \Pr[Y \leq w \mid X \geq x] \Pr[X \geq x]$$

The threshold x for variable X will be fixed later such that the probability $\Pr[X < x]$ is negligible. With this in mind, we simplify the expression above by upper-bounding it in the following way:

(7.17)
$$\Pr[Y \leq w] \leq \Pr[X < x] + \Pr[Y \leq w \mid X \geq x]$$

Note that decreasing X also makes Y smaller. This is to be understood in the following way: Y is dependent on X. For all $x_1 \leq x_2$ and for all y it holds that $\Pr[Y \leq y|X = x_1] \geq \Pr[Y \leq y|X = x_2]$. In other words, Y conditioned on $X = x_1$ is less than Y conditioned on $X = x_2$ in the *usual stochastic order*. Therefore:

(7.18)
$$\Pr[Y \leq w] \leq \Pr[X < x] + \Pr[Y \leq w \mid X = x]$$

We now fix the threshold x to $(m/n - \epsilon_1)t$ and use the fact that X is (n, t, m)-hypergeometrically distributed (of mean mt/n) to apply Corollary 2.1 (tail-bound for hypergeometric distributions) and obtain, for all $\epsilon_1 > 0$:

(7.19)
$$\Pr\left[X \leq \left(\frac{m}{n} - \epsilon_1\right) t\right] \leq \exp\left(-2\frac{t^2}{m}\epsilon_1^2\right) = \exp\left(-\frac{4\tau^2\epsilon_1^2}{1-2\phi}n\right)$$

In other words, this means that with high probability, the attack will affect at least $(m/n - \epsilon_1)\, t$ test runs. As the next step, we can derive from here a bound on the probability that the random variable Y is below some threshold. Since the type of test runs is sampled independently uniformly at random from the set of test runs, Y conditioned on the event that $X = (m/n - \epsilon_1)\, t$ follows a $((m/n - \epsilon_1)\, t, 1/k)$-binomial distribution. Let $\epsilon_2 > 0$. Applying Lemma 2.1 (Hoeffding's bound for binomial distributions), we arrive at:

$$\begin{aligned} \Pr\left[Y \leq \left(\frac{1}{k} - \epsilon_2\right) \left(\frac{m}{n} - \epsilon_1\right) t \,\middle|\, X = \left(\frac{m}{n} - \epsilon_1\right) t \right] &\leq \exp\left(-2t \left(\frac{m}{n} - \epsilon_1\right) \epsilon_2^2\right) \\ &= \exp\left(-\left(1 - 2\phi - 2\epsilon_1\right) \tau \epsilon_2^2 n\right) \end{aligned} \tag{7.20}$$

We choose to set the threshold for failed test runs to the value above $w = (1/k - \epsilon_2)\,(1/2 - \phi - \epsilon_1)\, t$. Combining the previous expressions, we therefore obtain:

$$\Pr\left[Y \leq w\right] \leq \exp\left(-\frac{4\tau^2 \epsilon_1^2}{1 - 2\phi} n\right) + \exp\left(-\left(1 - 2\phi - 2\epsilon_1\right) \tau \epsilon_2^2 n\right) \tag{7.21}$$

We combine this bound on failed test runs with the bound on affected computation runs (Eq. 7.15) to upper-bound the failure probability given in Eq. 7.14 for $m = (1/2 - \phi)n$ as follows:

$$\Pr\left[\text{fail}\right] \leq \max\left\{ \exp\left(-\frac{4\delta^2 \phi^2}{1 - 2\phi} n\right),\ \exp\left(-\frac{4\tau^2 \epsilon_1^2}{1 - 2\phi} n\right) + \exp\left(-\left(1 - 2\phi - 2\epsilon_1\right) \tau \epsilon_2^2 n\right) \right\} \tag{7.22}$$

This bound holds for any $\omega = (1/k - \epsilon_2)(1/2 - \phi - \epsilon_1)$, $\epsilon_1 \in (0, 1/2)$, $\epsilon_2 \in (0, 1/k)$ and $\phi \in (0, 1/2 - \epsilon_1)$. To obtain an optimal bound, this expression must be minimized over ϵ_1, ϵ_2, and ϕ. Irrespective of the exact form of the optimal bound, choosing ϕ, ϵ_1, and ϵ_2 sufficiently small implies the existence of protocols with verification exponential in n, for any fixed $0 < \omega < 1/2k$ and fixed $\delta, \tau \in (0, 1)$. ■

7.3.2 Noise Robustness

7.3.2.1 Local-Correctness on Honest-but-Noisy Devices

The local-correctness property discussed in the previous section did not take into account device imperfections. In fact, the analysis of blindness and verification makes no distinction between these imperfections and potentially malicious behaviours. Although satisfying these properties makes our protocol a concrete implementation of the Ideal Resource for Verifiable Delegated Quantum Computation, it could still fall short of expectations in terms of usability. Fortunately, for a class of realistic imperfections, our protocol has the additional property of being capable of correcting their impact and accepting with high probability. The final outcome is then the same as that obtained on noiseless devices with honest participants.

This additional *noise-robustness* property, the main innovation of this work, amounts to prove that Protocol 27 satisfies the local-correctness property with negligible ϵ_{cor} for noisy honest Client and/or Server devices. We prove this property under the following restrictions:

- The noise can be modelled by run-dependent Markovian processes – i.e. a possibly different arbitrary CPTP map acting on each run.

- The probability that at least one of the trap measurements fails in any single test run is upper-bounded by some constant $p_{max} < 1/2$ and lower-bounded by $p_{min} \leq p_{max}$.

Theorem 7.2 states that, in order for the protocol to terminate correctly with overwhelming probability on these noisy devices, w should be chosen such that $w/t > p_{max}$. Conversely, for any choice of $w/t < p_{min}$, we show that the protocol aborts with overwhelming probability. Recall that the constant ratios of test, computation and tolerated failed test runs are given by δ, τ and ω.

Theorem 7.2 (Local-Correctness of VDQC Protocol on Noisy Devices)**.** *Assume a Markovian run-dependent model for the noise on Client and Server devices and let $p_{min} \leq p_{max} < 1/2$ be respectively a lower and an upper-bound on the probability that at least one of the trap measurement outcomes in a single test run is incorrect. If $\omega > p_{max}$, Protocol 27 is ϵ_{cor}-locally-correct with exponentially-low ϵ_{cor}:*

$$\epsilon_{cor} = \exp\left(-2(\omega - p_{max})^2 \tau n\right) + \exp\left(-2\left(\frac{1}{2} - p_{max}\right)^2 \delta n\right). \tag{7.23}$$

On the other hand, if $\omega < p_{min}$, the Client's acceptance probability in Protocol 27 is exponentially-low and upper-bounded by:

$$\exp\left(-2(p_{min} - \omega)^2 \tau n\right) \tag{7.24}$$

Proof. We define random variables Y that corresponds to the number of failed test runs during one execution of the protocol, and Z counting the number of affected computation runs (where at least one bit of output is flipped). We call Ok the event that the Client accepts at the end of the protocol - if not many test runs fail, meaning that $Y < w$ - and $\mathsf{Correct}$ the event corresponding to a correct output - if few of the computation runs have their output bits flipped and therefore $Z < d/2$.

For $\boldsymbol{\omega > p_{max}}$. Equivalently, we have that $w > tp_{max}$. We are looking to lower-bound the probability of an honest run producing the correct outcome and not aborting. This happens if the noise has not disturbed too many computation and test runs (less than $d/2$ and w respectively):

$$\Pr\left[\mathsf{Correct} \wedge \mathsf{Ok}\right] = \Pr\left[Z < \frac{d}{2} \wedge Y < w\right] = \Pr\left[Z < \frac{d}{2}\right] \Pr\left[Y < w\right] \tag{7.25}$$

The second equality stems from the fact that the noise on runs is independent across runs and the nature of each run is chosen uniformly at random. We start by considering the effect on computation runs. If a computation run is affected by a given noise, then there is at least one type of test run that would have been affected (triggering traps) by the same noise. Since p_{max} is an upper-bound on the probability that any type of test run fails, then it is also an upper-bound on the probability that the outcome of the computation is incorrect if we assume (in the worst case) that any corruption on a single qubit of a computation run leads to a corrupted computation. Let $\hat{Z}_1$ be a random variable following a (d, p_{max})-binomial distribution of mean dp_{max}. Since we suppose that the noise is not correlated across

runs (meaning that the probabilities runs fail on noisy devices are independent), Z is upper-bounded by $\hat{Z}_1$ in the *usual stochastic order*, which in this case gives:

$$\Pr\left[Z < \frac{d}{2}\right] \geq \Pr\left[\hat{Z}_1 < \frac{d}{2}\right] = 1 - \Pr\left[\hat{Z}_1 \geq \frac{d}{2}\right] \tag{7.26}$$

Since $p_{max} < 1/2$ then $E\left(\hat{Z}_1\right) = dp_{max} < d/2$ and Lemma 2.1 yields:

$$\Pr\left[\hat{Z}_1 \geq \frac{d}{2}\right] \leq \exp\left(-2\frac{\left(dp_{max} - \frac{d}{2}\right)^2}{d}\right) = \exp\left(-2\delta\left(p_{max} - \frac{1}{2}\right)^2 n\right) = \epsilon_Z \in \text{negl}(n) \tag{7.27}$$

Then $\Pr[\mathsf{Correct}] = \Pr[Z < d/2] \geq 1 - \epsilon_Z$.

We can now focus on the test runs. Note that Y describes exactly the number of test rounds in which at least one trap measurement outcome is incorrect (by definition of a failed test run). The probability that a given test run fails is therefore upper-bounded by p_{max}. Let $\hat{Y}_1$ be a random variable following a (t, p_{max})-binomial distribution. Since we suppose that the noise is not correlated across runs, Y is upper-bounded by $\hat{Y}_1$ in the usual stochastic order:

$$\Pr[Y < w] \geq \Pr\left[\hat{Y}_1 < w\right] = 1 - \Pr\left[\hat{Y}_1 \geq w\right] \tag{7.28}$$

Further, since $E\left(\hat{Y}_1\right) = tp_{max} < w$, applying Lemma 2.1 yields:

$$\Pr\left[\hat{Y}_1 \geq w\right] \leq \exp\left(-2\frac{(tp_{max} - w)^2}{t}\right) = \exp\left(-2(\omega - p_{max})^2\tau n\right) = \epsilon_{Y,1} \tag{7.29}$$

Then $\Pr[\mathsf{Ok}] = \Pr[Y < w] \geq 1 - \Pr\left[\hat{Y}_1 \geq w\right] = 1 - \epsilon_{Y,1}$. We conclude that with high probability not enough test runs are corrupted to trigger an abort and the Client accepts the outcome of the computation.

Combining these inequalities gives:

$$\Pr[\mathsf{Correct} \wedge \mathsf{Ok}] \geq (1 - \epsilon_Z)(1 - \epsilon_{Y,1}) \geq 1 - (\epsilon_Z + \epsilon_{Y,1}) \tag{7.30}$$

For $\boldsymbol{\omega < p_{min}}$. In that case, we have that $tp_{min} > w$. We show that the probability of accepting is upper-bounded by a negligible function. Let $\hat{Y}_2$ be a random variable following a (t, p_{min})-binomial distribution, the number of failed test runs Y then is lower-bounded by $\hat{Y}_2$ of mean tp_{min} in the usual stochastic order:

$$\Pr[Y < w] \leq \Pr\left[\hat{Y}_2 < w\right] \tag{7.31}$$

This number of failed tests is higher than the acceptable threshold $tp_{min} > w$, therefore using Lemma 2.1 directly and the same simplifications as above, we get:

$$\Pr\left[\hat{Y}_2 < w\right] \leq \exp\left(-2(p_{min} - \omega)^2\tau n\right) = \epsilon_{Y,2} \tag{7.32}$$

Therefore $\Pr[\mathsf{Ok}] \leq \epsilon_{Y,2} \in \mathrm{negl}(n)$. This is a case of *false-positive* since the outcome is still correct in this scenario.

■

Since the results for blindness, blindness, input-independent verification and verifiability already integrate the noise in their analyses (as explained below, they consider the most general deviation, which includes noise), this new bound concerning local-correctness on noisy devices can also be used alongside these previous bounds using the Local-Reduction Theorem from [41], yielding in this case a value for $\epsilon = max\{\delta, \epsilon_{cor}\}$ that may now depend on the noise level of the devices since $\epsilon_{cor} > 0$ (however, note that if $\delta > \epsilon_{cor}$, then the noise has no impact on the total ϵ).

7.4 Conclusion and Discussion

We have presented above a protocol that is the first of its kind with respect to its ability to withstand noise while allowing to successfully execute computations in an unconditionally verifiable way with relatively modest overhead. These capabilities are made possible thanks to the nature of the computation, its deterministic classical output combined with a classical repetition code favourably replacing more resource-demanding fault-tolerant constructions. The obtained error-mitigation capabilities can then be used to tolerate noise while having the computation perform correctly.

Role of Noise Assumptions in Correctness Analysis. As stated in Section 7.3.1, our security proof does not rely on any assumption regarding a specific form or strength of the noise. On the contrary, it simply considers any deviation as potentially malicious and showed that the protocol would provide information theoretic verification and blindness of the computation.

The assumptions introduced Section 7.3.2, which deals with the correctness of the protocol on honest noisy devices, serve a different purpose: when the imperfections of the devices are light enough (markovian and of limited strength), we show their impact on the computation can always be mitigated. The noise models include not only independent Pauli operators acting on qubits, but also more general operators that are independent between various runs. As a consequence, under these mild restrictions, the computation will accept with high probability. If the Server sells to the Client a service with a low honest level of noise, the Client will hence not only get a security guarantee in case the Server lied and decides to deviate maliciously but also a performance guarantee that an honest computation will terminate correctly.

Consequently, there is no security risk in first probing the device to find out about the noise level and then using it to set the admissible ratio of failed test runs w/t to a compatible value. The value for w/t might even be adjusted between two executions of the full protocol to cope with drifting values of noise. An over-inflated value of w/t only results in superfluous repetitions and hence a longer running time for the protocol. Conversely, setting the value of w/t too low carries the risk of aborting most of the time and thus not being able to ever complete the computation, as is the case with previous protocols without error-correction capabilities.

Regarding the assumption that the noise maps between each run are independent, this can realistically be achieved in an experimental setup by simply waiting long enough between each run for all the states to decohere or resetting the state of the system. This guarantees that the noise of one run only depends

on the classical parameters of a previous run. This of course prolongs the duration of the experiment, but this overhead is not too prohibitive considering the low coherence time of quantum memories. We do not require that the noise is identically distributed, which takes into account situations where the noise level might fluctuate in an experiment (e.g. temperature changes, optical alignment modification, transient vibrations).

Decoupling Verifiability and Fault-Tolerance. We want to emphasize that our scheme does *not* rely on nor provide fault-tolerance. It is quite the opposite in that it decouples the quantum error-correction scheme devoted to fault-tolerant computing from the (in our case classical) one that ensures sensitive verification.

More precisely, traps inserted in MBQC schemes (see [53, 78]) have bounded sensitivity, i.e. the probability ϵ of not detecting an attack with a given trap is bounded away from 0. In these schemes, to boost the detection sensitivity of each trap exponentially, the computation path is encoded fault-tolerantly so that all small-weight errors can be corrected. Then, if the number of correctable errors is d then a malicious Server has to attack at least d locations for it to have a non-trivial effect on the computation. This, in turn, dramatically reduces the probability of a trap not detecting an effective attack to ϵ^d. This use of fault-tolerance is comparable to the repetition of test rounds in our protocol.

Quite naturally, the fault-tolerant encoding induces overhead that reduces the size of the computation that can be verified using a given finite-sized Server. But the real trouble comes from the realization that the magnitude of this overhead is dictated by the security objectives of the Client: the higher the desired trap sensitivity, the larger the overhead. Hence, while our protocol enables a Client to use all the available qubits to perform a verified computation up to an arbitrary security level, using the techniques from [53, 78] and requiring the same security level might simply consume all the qubits to build a single trusted qubit.

However, for our protocol to accept with overwhelming probability, the average ratio of failed test runs must be upper-bounded away from $1/2k$. This is unfortunately a global metric: it is easier to attain a perceived noise level below our $1/2k$ threshold for a 2-qubit computation than it is for a 100-qubit computation. This can be understood in the following way. The built-in error mitigation capability of our protocol does not stop errors from propagating into the result, but only serves to recover the correct result from the noisy outcomes of each unprotected computational run. The other way around, proper fault-tolerance prevents error-propagation and would be useful in lowering the ratio of failed test runs in large computations and arrive at the $1/2k$ ratio of failed test runs required to perform the verified computation. Anyhow, this would also be required for implementations of [53, 78]: the aforementioned fault-tolerant encoding only takes care of sensitivity boosting while leaving all the traps unprotected and vulnerable to noise.

To summarize, the advantage of the decoupling we propose between fault-tolerant computation and robust verification is that, while fault-tolerance is likely unavoidable in order to achieve the noise level for our protocol to be useful to honest participants for large computations on yet-to-be-created systems, it does not condition the verifiability bounds of our protocol whatsoever and our security parameters can be chosen independently of the quantum error-correction used to perform the computation. Conversely, those same bounds would be unreachable on realistic systems if the verification is performed using fault-tolerant trap sensitivity boosting due to large resource overheads imposed by even moderately

stringent security levels.

Fine-Tuning the Number of Repetitions. Once the noise level p_{max} has been determined and used to constrain w as explained in Section 7.3.2, we can look at ways to optimize the resource overhead for robust verification in terms of excess number of runs compared to standard MBQC[5].

The first obvious parameter influencing the overhead is the number k of different types of test runs. We can look at the problem the other way around. The amount of tolerated noise p_{max}, which is inherent to the devices upon which the computation is being performed, is upper-bounded by ω and the correctness bound on noisy devices depends exponentially on the square of the gap between these two values. On the other hand, the value of ω is upper-bounded by $1/2k$. Therefore, higher values of k induce lower amounts of tolerated noise and a higher number of repetitions. In our scheme, k also corresponds to the number of colours in the graph colouring chosen by the Client on the Server's graph G. While the problem of finding an optimal (i.e. minimal) graph colouring is NP–Complete for general graphs, there exist efficient algorithms to compute approximately optimal graph colourings.

For any general graph, a greedy algorithm yields a k-colouring for $k \leq D(G)+1$, with $D(G)$ being the maximum degree of G. Also note that most graphs used in MBQC are planar and the celebrated 4-Colour Theorem states that any planar graph is 4-colourable, hence bounding k by 4. Efficient algorithms to find such a colouring exist (quadratic in the number of vertices of the graph [120]) which would then be of practical interest in designing robust verifiable schemes. Furthermore, as $k = 2$ is the best possible value for our scheme and because it is efficient to check if a graph is bi-partite, the value $k = 2$ should be tested. Note that the brickwork graph (which is universal for quantum computations) and all dotted (edge-decorated) graphs are bipartite. In contrast, testing the case of a 3-colourable graph is NP–Complete, so for large planar graphs the Client may have to choose a 4-colouring instead at the expense of more repetitions of the protocol for attaining the same verification bound.

Once k is fixed, for targeted values of perceived noise level resistance, acceptance and failure probabilities, numerical optimizations can be used to determine the best values for the total number of runs n, the ratio of test runs t/n, and the ratio of test runs allowed to fail w/t using equation 7.22.

Link to Certification and Benchmarking. Finally, we want to point out a connection between certification and verification that stems directly from the presented protocol. As mentioned in the introduction, two broad strategies can be pursued when one wants to give guarantees with respect to a device sold to clients.

On one hand, the provider and the device can be certified by a trusted third party and the provider commits to manufacture the device that has been certified. The commitment is enforced not by design but legally. This is often chosen for efficiency reasons: the certification is done once and in case of widespread services or devices, the cost (in time, money, effort) to certify it is absorbed into the volume of service or devices provided. Another reason for choosing this form of certification is that it offers a natural way to cope with imperfect devices. Most devices are certified to within some acceptable range of performance describing its nominal behaviour.

[5]Note that this is the only source of overhead as each run requires the same resources as the original MBQC.

This however has caveats, as recent years have proven. For the commitment to be effective, there needs to be a reasonable chance to catch deviations from the certified behaviour which supposes in turn that devices are prevented from sensing whether they are being tested or not.[6]

On the other hand, the provider can choose to opt for a verifiable device (or service). In that case the security is better as there is no commitment required, only fact-based trust dependent on a series of tests. The trouble with such a strategy is - or rather was up to now - its high overhead in the context of quantum computing, making it inaccessible and extremely expensive in any foreseeable future.

Our results show that for certain classes of computations this does not need to be the case. In fact, the best of both worlds can be combined. Test runs are indeed probing whether our device and computation is abiding by some certification standards expressed in terms of an effective noise level. This is done continuously through the computation in order to prevent the device from adapting its behaviour. Since the computation is blind, even a fully malicious adversary cannot successfully fool the protocol. So in effect, blindness allows to combine computation and certification to arrive at verification, retaining the efficiency and imperfection tolerance of certification while keeping the unconditional security of verified schemes. While this is clearly more expensive than simple certification, this overhead should be acceptable for a wide range of practical situations. A natural open question, that we leave for further research, is whether this strategy can be extended to other protocols and if there are situations where other schemes are more efficient.

Future Work

As part of the Quantum Internet Alliance Blueprint Project which aims at demonstrating a full-stack implementation of a multi-node quantum protocol (from classical control using a generic quantum programming language all the way down to the hardware), we have had discussions for the past year with two experimental teams for the purpose of performing the protocol presented above in the simplest case (two Server-side qubits) using the setup presented in Figure 7.1.

We are currently trying to find optimisations of the bound presented in Equation 7.22 given fixed values for the total number of completed runs, as this number is directly proportionate to the time required for running the experiment.

Future improvements that are being considered as well look into expanding the range of feasible computations from almost-deterministic to BQP decision problems, and later sampling problems.

Another direction is to find a way to bridge even closer benchmarking and verification by looking at the possible computation that can be performed by a Server if the Client, upon accepting a non-corrupted computation, returns the secret parameters of the traps and indicates to the Server which ones were triggered. This could help the Server with improving its own architecture by acquiring valuable information regarding the operations that fail most often. If it doesn't break the security of the Client, it is a win-win situation and could potentially be an argument against the Server cheating. The type of information that the Server can gather depends on the types of test runs that are performed, so it is a

[6]This was exactly the strategy developed by some car makers: by guessing when the engine was being run on a test-bench they would reduce its power as to pass the tests while offering a widely different behaviour when in real conditions. It was only after independent associations measured the emissions in road-like conditions and found a gap large enough that it could not be explained by variations in physical conditions that extensive search was conducted and the deviation discovered. Had the gap been smaller, it would have most certainly gone undetected.

challenge to find the test that would yield the most information to the Server. Interweaving test types across different vertices could be a possible way to probe more interesting properties.

Lowering the Number of Different Test Runs. Finally, due to the importance of the value of k governing the types of test runs that can be performed, we are investigating ways in which this value can be reduced, for example by merging various vertices of different colours in a graph into a single vertex to lower the number of colours required. This merged vertex would then need to be tested differently from the current single-qubit vertices, by embedding deterministic Clifford computations (or any deterministic computation that can be simulated in a reasonable amount of time) in the graph in these locations unknown to the Server. The result of [20] indicates that at least two types of test runs may be required for these merged vertices. These computations are separated from the rest of the graph using the same dummy qubits as in the previous case. They could allow for adversarially-secure benchmarking of the hardware, where not even the hardware or its operator are aware of the tests being performed. This question is therefore tightly intertwined with the previous one and it seems like there might be a trade-off between the two goals. The idea presented above can be formalised as follows.

Definition 7.2 (Connected Open Graph Covering)**.** *Let $G = (V, E)$ be a finite undirected graph, $C = \{V_i\}_i$ with $V_i \subseteq V$ for all i and $F \subseteq V$. We say that $\{C, F\}$ is an* open connected covering *of graph G with frontier F if the following conditions are met (and call the V_i the components of the covering):*

1. *The V_i along with the frontier are a vertex-covering of graph G: $F \bigcup_i V_i = V$.*
2. *No two vertices of different components are linked via an edge: for all $i \neq j$, $v \in V_i$ and $w \in V_j$, we have $(v, w) \notin E$ and $(w, v) \notin E$. Equivalently, all exterior vertices of each component are only linked to frontier vertices: $\bigcup_{v \in V_i} N_G(v) \setminus V_i \subset F$, where $N_G(v)$ is the set of neighbours of vertex v in graph G.*
3. *Each sub-graph $G_i := (V_i, V_i \times V_i \cap E)$ is connected.*
4. *The frontier and components are disjoint: for all i, $V_i \cap F = \emptyset$; and the components are pair-wise disjoint: for all $i \neq j$, $V_i \cap V_j = \emptyset$.*

We say that such a covering is testable if it further satisfies the following definition:

Definition 7.3 (Deterministically Testable Covering)**.** *We say that a connected open covering of a graph G is* deterministically testable *if each component V_i of the covering has a flow $(f_i, \preceq_i)$ and it is efficient to sample an MBQC computation on V_i defined by $\left(I_i, O_i, \{\alpha(v)\}_{v \in V_i \setminus O_i}\right)$ with $I_i, O_i \subset V_i$ such that, after the computation has been performed on input state $|+\rangle^{\otimes \# I_i}$, the result of a measurement in basis $\{|+\rangle, |-\rangle\}$ is deterministic. We further say that these tests are* flow-compatible *with a given flow $(f, \preceq)$ on G if for all i the measurement orderings induced by $\preceq_i$ and $\preceq$ are compatible, i.e. for all $v, w \in V_i$ we have $v \preceq_i w$ if and only if $v \preceq w$.*

Note that this is an extension of the notion of test runs that have been presented previously, as a graph where every qubit is either a trap or a dummy qubit is an example of a testable connected open covering of the graph. It is then possible to extend the notion of graph colouring to these coverings in the following way.

Definition 7.4 (Puzzle-Piece Coverings)**.** *Let $\{C_l, F_l\}_{l\in[k]}$ be a set of coverings and their associated frontiers for a given graph G. We say that this set is a* puzzle *of graph G if, for all l, $\bigcup_{l'\neq l} C_{l'} = F_l$ and $C_{l'} \cap C_l = \emptyset$ for all $l' \neq l$.*

We can then use these puzzle coverings for testing the honesty of the Server in the same way as the colourings presented above. For each test run, the Client will sample such a covering C_l from this set of coverings uniformly at random, along with a deterministic MBQC computation for each $V_{i,l}$. Then, for all qubits in the frontier F_l of the covering it will send dummy qubits. On the other hand, the rest of the qubits will be sent in a state $\left|+_{\theta(v)}\right\rangle$ with $\theta(v) \in_R \Theta$ chosen uniformly at random. This has the effect of separating the components $V_{i,l}$ into independent graphs, on which the Server performs the computation following the instructions of the Client. As before, due to the hiding property of the UBQC Protocol upon which this extension is based, the Server has no idea whether the qubits that it measures are performing the Client's true computation or correspond to a test and it is therefore deterred from cheating lest it springs a trap. Now the number of tests is governed by the new value for k, which is potentially lower than then one resulting from the choice of colouring described earlier. However, more research is needed to precisely evaluate the number of test types required per covering to determine if there is any real gain in using this technique. It should be sufficient to define test runs such that no Pauli deviation of the Server commutes with all possible tests. Going further, in the case where the initial graph can be used to perform deterministic MBQC computations, the trivial puzzle-piece covering consisting of the whole graph could be considered as well and the number of test runs would simply consist of those required to test for any possible deviation on the entire graph.

The previous protocols only assume a number of qubits that is barely large enough to perform the initial desired computation of the Client. If we push the number of qubits even further, where it stops being prohibitive to run multiple different computations at the same time in parallel, we can instead embed the computation that is to be performed in one of the covering components V_i. Interestingly, the coverings can be seen as extensions of trap insertion techniques such as the ones from [53, 78]. The sampling of tests by the Client no longer rely on puzzle coverings but instead on choosing uniformly at random an open connected covering from a set of coverings such that one of the components always allows the initial computation to be performed. There is however a necessary condition on this sampling: every qubit must have a non-negligible probability of being included in one of the test components during this sampling. More work is required to determine whether this condition is sufficient to guarantee the verifiability of the overall scheme.

Replacing the Dotted-Triple Graph VBQC in MPQC Protocol. The extreme case where the whole graph is considered to be a single puzzle piece leads to an interesting situation: there is no need to use dummies to break this graph apart and the Client needs only to send $|+_\theta\rangle$ states to the Server. Recalling the proofs and discussions of the previous chapter, it is simple to produce collaboratively encrypted $|+_\theta\rangle$ states in such a way that the Server's deviation commutes with the preparation phase without picking up a dependency on the honest Client's secret parameters. The troublesome part stems only from the preparation of the dummies, which requires both a more in-depth analysis and multiply the number of qubits sent by the each Client by a factor 9. Removing the need to produce dummies in an MBQC-based verifiable blind protocol can drastically simplify the MPQC protocol previously

presented in this thesis while reducing the Server's memory requirement to $\#U + N$ where $\#U$ is the number of qubit in the base MBQC computation that the N Clients wish to run on the Server.

Bibliography

[1] E. Adlam and A. Kent, *Knowledge-concealing evidencing of knowledge about a quantum state*, Phys. Rev. Lett., 120 (2018), p. 050501, https://doi.org/10.1103/PhysRevLett.120.050501, https://link.aps.org/doi/10.1103/PhysRevLett.120.050501.

[2] D. Aharonov, A. Chailloux, M. Ganz, I. Kerenidis, and L. Magnin, *A simpler proof of the existence of quantum weak coin flipping with arbitrarily small bias*, SIAM Journal on Computing, 45 (2016), pp. 633–679, https://doi.org/10.1137/14096387X, https://doi.org/10.1137/14096387X.

[3] G. Alagic and A. Russell, *Quantum-secure symmetric-key cryptography based on hidden shifts*, in Advances in Cryptology – EUROCRYPT 2017, J.-S. Coron and J. B. Nielsen, eds., Cham, 2017, Springer International Publishing, pp. 65–93, https://doi.org/10.1007/978-3-319-56617-7_3, https://link.springer.com/chapter/10.1007/978-3-319-56617-7_3.

[4] P. Aliferis and D. W. Leung, *Computation by measurements: A unifying picture*, Phys. Rev. A, 70 (2004), p. 062314, https://doi.org/10.1103/PhysRevA.70.062314, http://link.aps.org/doi/10.1103/PhysRevA.70.062314.

[5] B. Alon, H. Chung, K.-M. Chung, M.-Y. Huang, Y. Lee, and Y.-C. Shen, *Round efficient secure multiparty quantum computation with identifiable abort*, 2020, https://eprint.iacr.org/2020/1464.

[6] A. Ambainis, *A new protocol and lower bounds for quantum coin flipping*, Journal of Computer and System Sciences, 68 (2004), pp. 398 – 416, https://doi.org/https://doi.org/10.1016/j.jcss.2003.07.010, http://www.sciencedirect.com/science/article/pii/S0022000003001417.
Special Issue on STOC 2001.

[7] A. Ambainis, A. Rosmanis, and D. Unruh, *Quantum attacks on classical proof systems: The hardness of quantum rewinding*, in 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, Oct 2014, pp. 474–483, https://doi.org/10.1109/FOCS.2014.57, https://www.computer.org/csdl/proceedings-article/focs/2014/6517a474/12OmNAHEpAr.

[8] M. V. Anand, E. E. Targhi, G. N. Tabia, and D. Unruh, *Post-quantum security of the cbc, cfb, ofb, ctr, and xts modes of operation*, in Post-Quantum Cryptography, T. Takagi, ed., Cham, 2016, Springer International Publishing, pp. 44–63, https://doi.org/10.1007/978-3-319-29360-8_4, https://link.springer.com/chapter/10.1007%2F978-3-319-29360-8_4.

[9] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, *Quantum supremacy using a programmable superconducting processor*, Nature, 574 (2019), pp. 505–510, https://doi.org/10.1038/s41586-019-1666-5, https://doi.org/10.1038/s41586-019-1666-5.

[10] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, *Multiparty computation with low communication, computation and interaction via threshold fhe*, in Advances in Cryptology – EUROCRYPT 2012, D. Pointcheval and T. Johansson, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 483–501, https://doi.org/10.1007/978-3-642-29011-4_29, https://link.springer.com/chapter/10.1007/978-3-642-29011-4_29.

[11] Y. Aumann and Y. Lindell, *Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 137–156, https://doi.org/10.1007/978-3-540-70936-7_8, http://dx.doi.org/10.1007/978-3-540-70936-7_8.

[12] S. Barz, J. F. Fitzsimons, E. Kashefi, and P. Walther, *Experimental verification of quantum computation*, Nature Physics, 9 (2013), pp. 727–731, https://doi.org/10.1038/nphys2763, https://doi.org/10.1038/nphys2763.

[13] S. Barz, E. Kashefi, A. Broadbent, J. F. Fitzsimons, A. Zeilinger, and P. Walther, *Demonstration of blind quantum computing*, Science, 335 (2012), pp. 303–308, https://doi.org/10.1126/science.1214707, https://science.sciencemag.org/content/335/6066/303, https://arxiv.org/abs/https://science.sciencemag.org/content/335/6066/303.full.pdf.

[14] M. Ben-Or, C. Crepeau, D. Gottesman, A. Hassidim, and A. Smith, *Secure multiparty quantum computation with (only) a strict honest majority*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, FOCS '06, Washington, DC, USA, 2006, IEEE Computer Society, pp. 249–260, https://doi.org/10.1109/FOCS.2006.68, http://dx.doi.org/10.1109/FOCS.2006.68.

[15] C. H. Bennett and G. Brassard, *Quantum cryptography: Public key distribution and coin tossing*, Theoretical Computer Science, 560 (2014), pp. 7 – 11,

https://doi.org/10.1016/j.tcs.2014.05.025, http://www.sciencedirect.com/science/article/pii/S0304397514004241.
Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84.

[16] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, *Threshold cryptosystems from threshold fully homomorphic encryption*, in Advances in Cryptology – CRYPTO 2018, H. Shacham and A. Boldyreva, eds., Cham, 2018, Springer International Publishing, pp. 565–596, https://doi.org/10.1007/978-3-319-96884-1_19, https://link.springer.com/chapter/10.1007%2F978-3-319-96884-1_19.

[17] D. Boneh and M. Zhandry, *Secure signatures and chosen ciphertext security in a quantum computing world*, in Advances in Cryptology – CRYPTO 2013, R. Canetti and J. A. Garay, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 361–379, https://doi.org/10.1007/978-3-642-40084-1_21, https://link.springer.com/chapter/10.1007%2F978-3-642-40084-1_21.

[18] X. Bonnetain, M. Naya-Plasencia, and A. Schrottenloher, *Quantum security analysis of aes*, IACR Transactions on Symmetric Cryptology, 2019 (2019), pp. 55–93, https://doi.org/10.13154/tosc.v2019.i2.55-93, https://tosc.iacr.org/index.php/ToSC/article/view/8314.

[19] G. Brassard, C. Crepeau, R. Jozsa, and D. Langlois, *A quantum bit commitment scheme provably unbreakable by both parties*, in Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science, 1993, pp. 362–371, https://doi.org/10.1109/SFCS.1993.366851, https://ieeexplore.ieee.org/abstract/document/366851.

[20] A. Broadbent, *How to verify a quantum computation*, Theory of Computing, 14 (2018), pp. 1–37, https://doi.org/10.4086/toc.2018.v014a011, http://www.theoryofcomputing.org/articles/v014a011.

[21] A. Broadbent, J. Fitzsimons, and E. Kashefi, *Measurement-Based and Universal Blind Quantum Computation*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 43–86, https://doi.org/10.1007/978-3-642-13678-8_2, https://doi.org/10.1007/978-3-642-13678-8_2.

[22] D. E. Browne, E. Kashefi, M. Mhalla, and S. Perdrix, *Generalized flow and determinism in measurement-based quantum computation*, New Journal of Physics, 9 (2007), p. 250, http://stacks.iop.org/1367-2630/9/i=8/a=250.

[23] H. Buhrman, N. Linden, L. Mančinska, A. Montanaro, and M. Ozols, *Quantum majority and other boolean functions with quantum inputs*, 2021, https://www.youtube.com/watch?v=0l49tmUimhk.

[24] N. Büscher, D. Demmler, N. P. Karvelas, S. Katzenbeisser, J. Krämer, D. Rathee, T. Schneider, and P. Struck, *Secure two-party computation in a quantum world*, in Applied Cryptography and Network Security, M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, eds., Cham, 2020, Springer International Publishing, pp. 461–480, https://doi.org/10.1007/978-3-030-57808-4_23, https://link.springer.com/chapter/10.1007/978-3-030-57808-4_23.

[25] R. Canetti, *Universally composable security: a new paradigm for cryptographic protocols*, in Proceedings 42nd IEEE Symposium on Foundations of Computer Science, 2001, pp. 136–145, https://doi.org/10.1109/SFCS.2001.959888.

[26] A. Chailloux and A. Leverrier, *Relativistic (or 2-prover 1-round) zero-knowledge protocol for* NP *secure against quantum adversaries*, in Advances in Cryptology – EUROCRYPT 2017, J.-S. Coron and J. B. Nielsen, eds., Cham, 2017, Springer International Publishing, pp. 369–396, https://doi.org/10.1007/978-3-319-56617-7_13, https://link.springer.com/chapter/10.1007%2F978-3-319-56617-7_13.

[27] A. Chi-Chih Yao, *Quantum circuit complexity*, in Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science, Nov 1993, pp. 352–361, https://doi.org/10.1109/SFCS.1993.366852, https://ieeexplore.ieee.org/document/366852.

[28] S. Coretti, U. Maurer, and B. Tackmann, *Constructing confidential channels from authenticated channels—public-key encryption revisited*, in Advances in Cryptology - ASIACRYPT 2013, K. Sako and P. Sarkar, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 134–153, https://doi.org/10.1007/978-3-642-42033-7_8, https://link.springer.com/chapter/10.1007/978-3-642-42033-7_8.

[29] R. Cramer, I. B. Damgrd, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, Cambridge University Press, USA, 1st ed., 2015, https://dl.acm.org/doi/book/10.5555/2846411.

[30] C. Crépeau, D. Gottesman, and A. Smith, *Secure multi-party quantum computation*, in Proceedings of the Thiry-fourth Annual ACM Symp. on Theory of Computing, STOC '02, New York, NY, USA, 2002, ACM, p. 643, https://doi.org/10.1145/509907.510000, http://doi.acm.org/10.1145/509907.510000.

[31] C. Crepeau and J. Kilian, *Achieving oblivious transfer using weakened security assumptions*, in [Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science, 1988, pp. 42–52, https://doi.org/10.1109/SFCS.1988.21920, https://ieeexplore.ieee.org/document/21920.

[32] J. Daemen and V. Rijmen, *Specification of Rijndael*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 31–51, https://doi.org/10.1007/978-3-662-04722-4_3, https://doi.org/10.1007/978-3-662-04722-4_3.

[33] I. Damgård, J. Funder, J. B. Nielsen, and L. Salvail, *Superposition attacks on cryptographic protocols*, in Information Theoretic Security, C. Padró, ed., Cham, 2014, Springer International Publishing, pp. 142–161, https://doi.org/10.1007/978-3-319-04268-8_9, https://link.springer.com/chapter/10.1007%2F978-3-319-04268-8_9.

[34] C. Dankert, R. Cleve, J. Emerson, and E. Livine, *Exact and approximate unitary 2-designs and their application to fidelity estimation*, Phys. Rev. A, 80 (2009), p. 012304, https://doi.org/10.1103/PhysRevA.80.012304, https://link.aps.org/doi/10.1103/PhysRevA.80.012304.

[35] V. Danos and E. Kashefi, *Determinism in the one-way model*, Phys. Rev. A, 74 (2006), p. 052310, https://doi.org/10.1103/PhysRevA.74.052310, http://link.aps.org/doi/10.1103/PhysRevA.74.052310.

[36] V. Danos, E. Kashefi, and P. Panangaden, *The measurement calculus*, J. ACM, 54 (2007), https://doi.org/10.1145/1219092.1219096, http://doi.acm.org/10.1145/1219092.1219096.

[37] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, 439 (1992), pp. 553–558, https://doi.org/10.1098/rspa.1992.0167, https://royalsocietypublishing.org/doi/10.1098/rspa.1992.0167.

[38] J. Don, S. Fehr, C. Majenz, and C. Schaffner, *Security of the fiat-shamir transformation in the quantum random-oracle model*, in Advances in Cryptology – CRYPTO 2019, A. Boldyreva and D. Micciancio, eds., Cham, 2019, Springer International Publishing, pp. 356–383, https://doi.org/10.1007%2F978-3-030-26951-7_13, https://link.springer.com/chapter/10.1007%2F978-3-030-26951-7_13.

[39] M. Doosti, N. Kumar, M. Delavar, and E. Kashefi, *Client-server identification protocols with quantum puf*, 2020, https://arxiv.org/abs/2006.04522.

[40] Y. Dulek, A. B. Grilo, S. Jeffery, C. Majenz, and C. Schaffner, *Secure multiparty quantum computation with a dishonest majority*, in Advances in Cryptology – EUROCRYPT 2020, A. Canteaut and Y. Ishai, eds., Cham, 2020, Springer International Publishing, pp. 729–758, https://doi.org/10.1007/978-3-030-45727-3_25, https://link.springer.com/chapter/10.1007/978-3-030-45727-3_25.

[41] V. Dunjko, J. F. Fitzsimons, C. Portmann, and R. Renner, *Composable security of delegated quantum computation*, in Advances in Cryptology – ASIACRYPT 2014, P. Sarkar and T. Iwata, eds., Berlin, Heidelberg, 2014, Springer Berlin Heidelberg, pp. 406–425, https://doi.org/10.1007/978-3-662-45608-8_22, https://link.springer.com/chapter/10.1007%2F978-3-662-45608-8_22.

[42] F. Dupuis, S. Fehr, P. Lamontagne, and L. Salvail, *Adaptive versus non-adaptive strategies in the quantum setting with applications*, in Advances in Cryptology – CRYPTO 2016, M. Robshaw and J. Katz, eds., Berlin, Heidelberg, 2016, Springer Berlin Heidelberg, pp. 33–59, https://doi.org/10.1007/978-3-662-53015-3_2, https://link.springer.com/chapter/10.1007/978-3-662-53015-3_2.

[43] F. Dupuis, S. Fehr, P. Lamontagne, and L. Salvail, *Secure certification of mixed quantum states with application to two-party randomness generation*, in Theory of Cryptography, A. Beimel and S. Dziembowski, eds., Cham, 2018, Springer International Publishing, pp. 282–314, https://doi.org/10.1007/978-3-030-03810-6_11, https://rd.springer.com/chapter/10.1007/978-3-030-03810-6_11.

[44] F. Dupuis, J. B. Nielsen, and L. Salvail, *Secure Two-Party Quantum Evaluation of Unitaries against Specious Adversaries*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010,

pp. 685–706, https://doi.org/10.1007/978-3-642-14623-7_37, http://dx.doi.org/10.1007/978-3-642-14623-7_37.

[45] F. Dupuis, J. B. Nielsen, and L. Salvail, *Actively Secure Two-Party Evaluation of Any Quantum Operation*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 794–811, https://doi.org/10.1007/978-3-642-32009-5_46, https://doi.org/10.1007/978-3-642-32009-5_46.

[46] S. Dziembowski and U. Maurer, *On generating the initial key in the bounded-storage model*, in Advances in Cryptology - EUROCRYPT 2004, C. Cachin and J. L. Camenisch, eds., Berlin, Heidelberg, 2004, Springer Berlin Heidelberg, pp. 126–137, https://doi.org/10.1007/978-3-540-24676-3_8, https://link.springer.com/chapter/10.1007%2F978-3-540-24676-3_8.

[47] P. H. Eberhard and R. R. Ross, *Quantum field theory cannot provide faster-than-light communication*, Foundations of Physics Letters, 2 (1989), pp. 127–149, https://doi.org/10.1007/BF00696109, https://link.springer.com/article/10.1007/BF00696109.

[48] A. K. Ekert, *Quantum cryptography based on bell's theorem*, Phys. Rev. Lett., 67 (1991), pp. 661–663, https://doi.org/10.1103/PhysRevLett.67.661, https://link.aps.org/doi/10.1103/PhysRevLett.67.661.

[49] S. Fehr, J. Katz, F. Song, H.-S. Zhou, and V. Zikas, *Feasibility and completeness of cryptographic tasks in the quantum world*, in Theory of Cryptography, A. Sahai, ed., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 281–296, https://doi.org/10.1007/978-3-642-36594-2_16, https://link.springer.com/chapter/10.1007/978-3-642-36594-2_16.

[50] W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, 1991, https://www.wiley.com/en-us/An+Introduction+to+Probability+Theory+and+Its+Applications%2C+Volume+2%2C+2nd+Edition-p-9780471257097.

[51] A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, in Advances in Cryptology — CRYPTO' 86, A. M. Odlyzko, ed., Berlin, Heidelberg, 1987, Springer Berlin Heidelberg, pp. 186–194, https://doi.org/10.1007/3-540-47721-7_12, https://link.springer.com/chapter/10.1007%2F3-540-47721-7_12.

[52] J. F. Fitzsimons, *Private quantum computation: an introduction to blind quantum computing and related protocols*, npj Quantum Information, 3 (2017), p. 23, https://doi.org/10.1038/s41534-017-0025-3, https://doi.org/10.1038/s41534-017-0025-3.

[53] J. F. Fitzsimons and E. Kashefi, *Unconditionally verifiable blind quantum computation*, Phys. Rev. A, 96 (2017), p. 012303, https://doi.org/10.1103/PhysRevA.96.012303, https://link.aps.org/doi/10.1103/PhysRevA.96.012303.

[54] T. Gagliardoni, A. Hülsing, and C. Schaffner, *Semantic security and indistinguishability in the quantum world*, in Advances in Cryptology – CRYPTO 2016, M. Robshaw and J. Katz, eds., Berlin, Heidelberg, 2016, Springer Berlin Heidelberg, pp. 60–89, https://doi.org/10.1007/978-3-662-53015-3_3, https://link.springer.com/chapter/10.1007/978-3-662-53015-3_3.

[55] R. Gennaro, C. Gentry, and B. Parno, *Non-interactive verifiable computing: Outsourcing computation to untrusted workers*, in Advances in Cryptology – CRYPTO 2010, T. Rabin, ed., Berlin, Heidelberg, 2010, Springer Berlin Heidelberg, pp. 465–482, https://doi.org/10.1007/978-3-642-14623-7_25, https://link.springer.com/chapter/10.1007/978-3-642-14623-7_25.

[56] I. Gerhardt, Q. Liu, A. Lamas-Linares, J. Skaar, C. Kurtsiefer, and V. Makarov, *Full-field implementation of a perfect eavesdropper on a quantum cryptography system*, Nature Communications, 2 (2011), p. 349, https://doi.org/10.1038/ncomms1348, https://doi.org/10.1038/ncomms1348.

[57] A. Gheorghiu, M. J. Hoban, and E. Kashefi, *A simple protocol for fault tolerant verification of quantum computation*, Quantum Science and Technology, 4 (2018), p. 015009, https://doi.org/10.1088/2058-9565/aaeeb3, https://doi.org/10.1088%2F2058-9565%2Faaeeb3.

[58] A. Gheorghiu, T. Kapourniotis, and E. Kashefi, *Verification of quantum computation: An overview of existing approaches*, Theory of Computing Systems, 63 (2019), pp. 715–808, https://doi.org/10.1007/s00224-018-9872-3, https://doi.org/10.1007/s00224-018-9872-3.

[59] G. C. Ghirardi, R. Grassi, A. Rimini, and T. Weber, *Experiments of the EPR type involving CP-violation do not allow faster-than-light communication between distant observers*, Europhysics Letters (EPL), 6 (1988), pp. 95–100, https://doi.org/10.1209/0295-5075/6/2/001, https://doi.org/10.1209/0295-5075/6/2/001.

[60] O. Goldreich, *Foundations of Cryptography*, vol. 1, Cambridge University Press, Cambridge, 2004, ch. Pseudorandom Permutations, pp. 164–169, https://doi.org/10.1017/CBO9780511721656.

[61] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game*, in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87, New York, NY, USA, 1987, ACM, pp. 218–229, https://doi.org/10.1145/28395.28420, http://doi.acm.org/10.1145/28395.28420.

[62] E. Greene and J. A. Wellner, *Exponential bounds for the hypergeometric distribution*, Bernoulli, 23 (2017), p. 1911–1950, https://doi.org/10.3150/15-bej800, http://dx.doi.org/10.3150/15-BEJ800.

[63] C. Greganti, M.-C. Roehsner, S. Barz, T. Morimae, , and P. Walther, *Demonstration of measurement-only blind quantum computing*, New Journal of Physics, 18 (2016), p. 250, https://doi.org/10.1088/1367-2630/18/1/013020, https://iopscience.iop.org/article/10.1088/1367-2630/18/1/013020.

[64] L. K. Grover, *Quantum mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett., 79 (1997), pp. 325–328, https://doi.org/10.1103/PhysRevLett.79.325, https://link.aps.org/doi/10.1103/PhysRevLett.79.325.

[65] S. Halevi, C. Hazay, A. Polychroniadou, and M. Venkitasubramaniam, *Round-optimal secure multi-party computation*, in Advances in Cryptology – CRYPTO 2018, H. Shacham and A. Boldyreva, eds., Cham, 2018, Springer International Publishing,

pp. 488–520, https://doi.org/10.1007/978-3-319-96881-0_17, https://link.springer.com/chapter/10.1007%2F978-3-319-96881-0_17.

[66] S. Hallgren, A. Smith, and F. Song, *Classical cryptographic protocols in a quantum world*, International Journal of Quantum Information, 13 (2015), p. 1550028, https://doi.org/10.1142/S0219749915500288, https://www.worldscientific.com/doi/abs/10.1142/S0219749915500288.

[67] M. Hayashi and T. Morimae, *Verifiable measurement-only blind quantum computing with stabilizer testing*, Phys. Rev. Lett., 115 (2015), p. 220502, https://doi.org/10.1103/PhysRevLett.115.220502, https://link.aps.org/doi/10.1103/PhysRevLett.115.220502.

[68] M. Hein, J. Eisert, and H. J. Briegel, *Multiparty entanglement in graph states*, Phys. Rev. A, 69 (2004), p. 062311, https://doi.org/10.1103/PhysRevA.69.062311, https://link.aps.org/doi/10.1103/PhysRevA.69.062311.

[69] M. Houshmand, M. Houshmand, S.-H. Tan, and J. Fitzsimons, *Composable secure multi-client delegated quantum computation*, arXiv e-prints, (2018), arXiv:1811.11929, https://arxiv.org/abs/1811.11929.

[70] M. Kaplan, G. Leurent, A. Leverrier, and M. Naya-Plasencia, *Breaking symmetric cryptosystems using quantum period finding*, in Advances in Cryptology – CRYPTO 2016, M. Robshaw and J. Katz, eds., Berlin, Heidelberg, 2016, Springer Berlin Heidelberg, pp. 207–237, https://doi.org/10.1007/978-3-662-53008-5_8, https://link.springer.com/chapter/10.1007/978-3-662-53008-5_8.

[71] T. Kapourniotis and A. Datta, *Nonadaptive fault-tolerant verification of quantum supremacy with noise*, Quantum, 3 (2019), p. 164, https://doi.org/10.22331/q-2019-07-12-164, https://doi.org/10.22331/q-2019-07-12-164.

[72] T. Kapourniotis, V. Dunjko, and E. Kashefi, *On optimising quantum communication in verifiable quantum computing*, arXiv e-prints, (2015), arXiv:1506.06943, https://arxiv.org/abs/1506.06943.

[73] T. Kapourniotis, E. Kashefi, L. Music, and H. Ollivier, *Delegating multi-party quantum computations vs. dishonest majority in two quantum rounds*, 2021, https://arxiv.org/abs/2102.12949.

[74] E. Kashefi, A. Kent, V. Vedral, and K. Banaszek, *Comparison of quantum oracles*, Phys. Rev. A, 65 (2002), p. 050304, https://doi.org/10.1103/PhysRevA.65.050304, https://link.aps.org/doi/10.1103/PhysRevA.65.050304.

[75] E. Kashefi, L. Music, and P. Wallden, *The quantum cut-and-choose technique and quantum two-party computation*, 2017, https://arxiv.org/abs/1703.03754, https://arxiv.org/abs/1703.03754.

[76] E. Kashefi and A. Pappa, *Multiparty delegated quantum computing*, Cryptography, 1 (2017), pp. 1–20, https://doi.org/10.3390/cryptography1020012, https://www.mdpi.com/2410-387X/1/2/12.

[77] E. Kashefi and P. Wallden, *Garbled quantum computation*, Cryptography, 1 (2017), https://doi.org/10.3390/cryptography1010006, https://www.mdpi.com/2410-387X/1/1/6.

[78] E. Kashefi and P. Wallden, *Optimised resource construction for verifiable quantum computation*, Journal of Physics A: Mathematical and Theoretical, 50 (2017), p. 145306, https://doi.org/10.1088/1751-8121/aa5dac, https://doi.org/10.1088/1751-8121/aa5dac.

[79] J. Kilian, *Founding crytpography on oblivious transfer*, in Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC '88, New York, NY, USA, 1988, Association for Computing Machinery, p. 20–31, https://doi.org/10.1145/62212.62215, https://doi.org/10.1145/62212.62215.

[80] M. S. Kiraz, *Secure and Fair Two-Party Computation*, PhD thesis, Department of Mathematics and Computer Science, 08 2008, https://doi.org/10.13140/RG.2.1.4458.2004.

[81] A. Y. Kitaev, *Quantum computations: algorithms and error correction*, Russian Mathematical Surveys, 52 (1997), pp. 1191–1249, https://doi.org/10.1070/rm1997v052n06abeh002155, https://doi.org/10.1070/rm1997v052n06abeh002155.

[82] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, *Randomized benchmarking of quantum gates*, Phys. Rev. A, 77 (2008), p. 012307, https://doi.org/10.1103/PhysRevA.77.012307, https://link.aps.org/doi/10.1103/PhysRevA.77.012307.

[83] V. Kolesnikov and T. Schneider, *Improved garbled circuit: Free xor gates and applications*, in Automata, Languages and Programming, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, eds., Berlin, Heidelberg, 2008, Springer Berlin Heidelberg, pp. 486–498, https://doi.org/10.1007/978-3-540-70583-3_40, https://link.springer.com/chapter/10.1007%2F978-3-540-70583-3_40.

[84] R. König, R. Renner, A. Bariska, and U. Maurer, *Small accessible quantum information does not imply security*, Phys. Rev. Lett., 98 (2007), p. 140502, https://doi.org/10.1103/PhysRevLett.98.140502, https://link.aps.org/doi/10.1103/PhysRevLett.98.140502.

[85] H. Kuwakado and M. Morii, *Quantum distinguisher between the 3-round feistel cipher and the random permutation*, in 2010 IEEE International Symposium on Information Theory, June 2010, pp. 2682–2685, https://doi.org/10.1109/ISIT.2010.5513654, https://ieeexplore.ieee.org/document/5513654.

[86] H. Kuwakado and M. Morii, *Security on the quantum-type even-mansour cipher*, in 2012 International Symposium on Information Theory and its Applications, Oct 2012, pp. 312–316, https://ieeexplore.ieee.org/document/6400943.

[87] D. Leichtle, L. Music, E. Kashefi, and H. Ollivier, *Verifying quantum computations on noisy devices with minimal overhead*, 2020, https://arxiv.org/abs/2011.10005, https://arxiv.org/abs/2011.10005.

[88] Y. Lindell and B. Pinkas, *An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries*, Springer, Berlin, Heidelberg, 2007, p. 52, https://doi.org/10.1007/978-3-540-72540-4_4, http://dx.doi.org/10.1007/978-3-540-72540-4_4.

[89] Y. Lindell and B. Pinkas, *A proof of security of yao's protocol for two-party computation*, Journal of Cryptology, 22 (2009), pp. 161–188, https://doi.org/10.1007/s00145-008-9036-8, https://doi.org/10.1007/s00145-008-9036-8.

[90] V. Lipinska, J. Ribeiro, and S. Wehner, *Secure multiparty quantum computation with few qubits*, Phys. Rev. A, 102 (2020), p. 022405, https://doi.org/10.1103/PhysRevA.102.022405, https://link.aps.org/doi/10.1103/PhysRevA.102.022405.

[91] M. Liu, J. Krämer, Y. Hu, and J. A. Buchmann, *Quantum security analysis of a lattice-based oblivious transfer protocol*, Frontiers Inf. Technol. Electron. Eng., 18 (2017), pp. 1348–1369, https://doi.org/10.1631/FITEE.1700039, https://doi.org/10.1631/FITEE.1700039.

[92] Q. Liu and M. Zhandry, *Revisiting post-quantum fiat-shamir*, in Advances in Cryptology – CRYPTO 2019, A. Boldyreva and D. Micciancio, eds., Cham, 2019, Springer International Publishing, pp. 326–355, https://doi.org/10.1007/978-3-030-26951-7_12, https://link.springer.com/chapter/10.1007%2F978-3-030-26951-7_12.

[93] H.-K. Lo, *Insecurity of quantum secure computations*, Phys. Rev. A, 56 (1997), pp. 1154–1162, https://doi.org/10.1103/PhysRevA.56.1154, http://link.aps.org/doi/10.1103/PhysRevA.56.1154.

[94] H.-K. Lo, *Insecurity of quantum secure computations*, Physical Review A, 56 (1997), p. 1154–1162, https://doi.org/10.1103/physreva.56.1154, http://dx.doi.org/10.1103/PhysRevA.56.1154.

[95] H.-K. Lo and H. F. Chau, *Is quantum bit commitment really possible?*, Phys. Rev. Lett., 78 (1997), pp. 3410–3413, https://doi.org/10.1103/PhysRevLett.78.3410, http://link.aps.org/doi/10.1103/PhysRevLett.78.3410.

[96] U. Mahadev, *Classical verification of quantum computations*, in 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, M. Thorup, ed., IEEE Computer Society, 2018, pp. 259–267, https://doi.org/10.1109/FOCS.2018.00033, https://doi.org/10.1109/FOCS.2018.00033.

[97] A. Mantri, T. F. Demarie, N. C. Menicucci, and J. F. Fitzsimons, *Flow ambiguity: A path towards classically driven blind quantum computation*, Phys. Rev. X, 7 (2017), p. 031004, https://doi.org/10.1103/PhysRevX.7.031004, https://link.aps.org/doi/10.1103/PhysRevX.7.031004.

[98] U. Maurer, *Constructive cryptography – a new paradigm for security definitions and proofs*, in Theory of Security and Applications, S. Mödersheim and C. Palamidessi, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 33–56, https://doi.org/10.1007/978-3-642-27375-9_3, https://link.springer.com/chapter/10.1007/978-3-642-27375-9_3.

[99] U. Maurer and R. Renner, *Abstract cryptography*, in Innovations in Computer Science, Tsinghua University Press, jan 2011, pp. 1 – 21, https://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/14.html.

[100] D. Mayers, *Unconditionally secure quantum bit commitment is impossible*, Phys. Rev. Lett., 78 (1997), pp. 3414–3417, https://doi.org/10.1103/PhysRevLett.78.3414, http://link.aps.org/doi/10.1103/PhysRevLett.78.3414.

[101] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. D. Morris, T. S. Humble, and R. C. Pooser, *Quantum chemistry as a benchmark for near-term quantum computers*,

npj Quantum Information, 5 (2019), p. 99, https://doi.org/10.1038/s41534-019-0209-0, https://doi.org/10.1038/s41534-019-0209-0.

[102] W. McCutcheon, A. Pappa, B. A. Bell, A. McMillan, A. Chailloux, T. Lawson, M. Mafu, D. Markham, E. Diamanti, I. Kerenidis, J. G. Rarity, and M. S. Tame, *Experimental verification of multipartite entanglement in quantum networks*, Nature Communications, 7 (2016), p. 13251, https://doi.org/10.1038/ncomms13251, https://doi.org/10.1038/ncomms13251.

[103] C. Mochon, *Quantum weak coin flipping with arbitrarily small bias*, 2007, https://arxiv.org/abs/0711.4114.

[104] T. Morimae and K. Fujii, *Secure entanglement distillation for double-server blind quantum computation*, Phys. Rev. Lett., 111 (2013), p. 020502, https://doi.org/10.1103/PhysRevLett.111.020502, https://link.aps.org/doi/10.1103/PhysRevLett.111.020502.

[105] S. Mossayebi and R. Schack, *Concrete Security Against Adversaries with Quantum Superposition Access to Encryption and Decryption Oracles*, arXiv e-prints, (2016), arXiv:1609.03780, https://arxiv.org/abs/1609.03780.

[106] L. Music, C. Chevalier, and E. Kashefi, *Dispelling myths on superposition attacks: Formal security model and attack analyses*, in Provable and Practical Security, K. Nguyen, W. Wu, K. Y. Lam, and H. Wang, eds., Cham, 2020, Springer International Publishing, pp. 318–337, https://doi.org/10.1007/978-3-030-62576-4_16, https://link.springer.com/chapter/10.1007/978-3-030-62576-4_16.

[107] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2000, https://doi.org/10.1017/CBO9780511976667.

[108] G. M. Nikolopoulos and E. Diamanti, *Continuous-variable quantum authentication of physical unclonable keys*, Scientific Reports, 7 (2017), p. 46047, https://doi.org/10.1038/srep46047, https://doi.org/10.1038/srep46047.

[109] N. None, *From long-distance entanglement to building a nationwide quantum internet: Report of the doe quantum internet blueprint workshop*, DOE OSTI Technical Report, (2020), https://doi.org/10.2172/1638794, https://www.osti.gov/biblio/1638794.

[110] A. Pappa, P. Jouguet, T. Lawson, A. Chailloux, M. Legré, P. Trinkler, I. Kerenidis, and E. Diamanti, *Experimental plug and play quantum coin flipping*, Nature Communications, 5 (2014), p. 3717, https://doi.org/10.1038/ncomms4717, https://doi.org/10.1038/ncomms4717.

[111] J. L. Park, *The concept of transition in quantum mechanics*, Foundations of Physics, 1 (1970), pp. 23–33, https://doi.org/10.1007/BF00708652, https://doi.org/10.1007/BF00708652.

[112] C. Peikert, V. Vaikuntanathan, and B. Waters, *A framework for efficient and composable oblivious transfer*, in Advances in Cryptology – CRYPTO 2008, D. Wagner, ed., Berlin, Heidelberg, 2008, Springer Berlin Heidelberg, pp. 554–

571, https://doi.org/10.1007/978-3-540-85174-5_31, https://link.springer.com/chapter/10.1007/978-3-540-85174-5_31.

[113] D. Pointcheval and J. Stern, *Security proofs for signature schemes*, in Advances in Cryptology — EUROCRYPT '96, U. Maurer, ed., Berlin, Heidelberg, 1996, Springer Berlin Heidelberg, pp. 387–398, https://doi.org/10.1007/3-540-68339-9_33, https://link.springer.com/chapter/10.1007%2F3-540-68339-9_33.

[114] C. Portmann, *Quantum authentication with key recycling*, in Advances in Cryptology – EUROCRYPT 2017, Proceedings, Part III, vol. 10212 of Lecture Notes in Computer Science, Springer, 2017, pp. 339–368, https://doi.org/10.1007/978-3-319-56617-7_12, https://link.springer.com/chapter/10.1007/978-3-319-56617-7_12.

[115] C. Portmann and R. Renner, *Cryptographic security of quantum key distribution*, arXiv e-prints, (2014), arXiv:1409.3525, https://arxiv.org/abs/1409.3525.

[116] J. Preskill, *Quantum Computing in the NISQ era and beyond*, Quantum, 2 (2018), p. 79, https://doi.org/10.22331/q-2018-08-06-79, https://doi.org/10.22331/q-2018-08-06-79.

[117] E. M. Rains, *Nonbinary quantum codes*, IEEE Transactions on Information Theory, 45 (1999), pp. 1827–1832, https://doi.org/10.1109/18.782103, https://ieeexplore.ieee.org/document/782103.

[118] R. Raussendorf and H. J. Briegel, *A one-way quantum computer*, Phys. Rev. Lett., 86 (2001), pp. 5188–5191, https://doi.org/10.1103/PhysRevLett.86.5188, http://link.aps.org/doi/10.1103/PhysRevLett.86.5188.

[119] H. Robbins, *A remark on stirling's formula*, The American Mathematical Monthly, 62 (1955), pp. 26–29, http://www.jstor.org/stable/2308012.

[120] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas, *Efficiently four-coloring planar graphs*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, New York, NY, USA, 1996, Association for Computing Machinery, p. 571–575, https://doi.org/10.1145/237814.238005, https://doi.org/10.1145/237814.238005.

[121] L. Salvail, C. Schaffner, and M. Sotáková, *On the Power of Two-Party Quantum Cryptography*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 70–87, https://doi.org/10.1007/978-3-642-10366-7_5, http://dx.doi.org/10.1007/978-3-642-10366-7_5.

[122] L. Salvail, C. Schaffner, and M. Sotáková, *Quantifying the leakage of quantum protocols for classical two-party cryptography*, International Journal of Quantum Information, 13 (2015), p. 1450041, https://doi.org/10.1142/S0219749914500415, https://doi.org/10.1142/S0219749914500415.

[123] R. J. Serfling, *Probability inequalities for the sum in sampling without replacement*, Ann. Statist., 2 (1974), pp. 39–48, https://doi.org/10.1214/aos/1176342611, https://doi.org/10.1214/aos/1176342611.

[124] P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, SFCS '94, USA, 1994, IEEE Computer Society, p. 124–134, https://doi.org/10.1109/SFCS.1994.365700, https://doi.org/10.1109/SFCS.1994.365700.

[125] B. Škorić, *Quantum readout of physical unclonable functions*, International Journal of Quantum Information, 10 (2012), p. 1250001, https://doi.org/10.1142/S0219749912500013, https://doi.org/10.1142/S0219749912500013.

[126] D. Unruh, *Universally composable quantum multi-party computation*, in Advances in Cryptology – EUROCRYPT 2010, H. Gilbert, ed., Berlin, Heidelberg, 2010, Springer Berlin Heidelberg, pp. 486–505, https://doi.org/10.1007/978-3-642-13190-5_25, https://link.springer.com/chapter/10.1007%2F978-3-642-13190-5_25.

[127] D. Unruh, *Quantum Proofs of Knowledge*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 135–152, https://doi.org/10.1007/978-3-642-29011-4_10, http://dx.doi.org/10.1007/978-3-642-29011-4_10.

[128] D. Unruh, *Collapse-binding quantum commitments without random oracles*, in Advances in Cryptology – ASIACRYPT 2016, J. H. Cheon and T. Takagi, eds., Berlin, Heidelberg, 2016, Springer Berlin Heidelberg, pp. 166–195, https://doi.org/10.1007/978-3-662-53890-6_6, https://link.springer.com/chapter/10.1007/978-3-662-53890-6_6.

[129] D. Unruh, *Computationally binding quantum commitments*, in Advances in Cryptology – EUROCRYPT 2016, M. Fischlin and J.-S. Coron, eds., Berlin, Heidelberg, 2016, Springer Berlin Heidelberg, pp. 497–527, https://doi.org/10.1007/978-3-662-49896-5_18, https://link.springer.com/chapter/10.1007/978-3-662-49896-5_18.

[130] J. Watrous, *Zero-knowledge against quantum attacks*, SIAM Journal on Computing, 39 (2009), pp. 25–58, https://doi.org/10.1137/060670997, http://dx.doi.org/10.1137/060670997.

[131] S. Wehner, C. Schaffner, and B. M. Terhal, *Cryptography from noisy storage*, Phys. Rev. Lett., 100 (2008), p. 220502, https://doi.org/10.1103/PhysRevLett.100.220502, https://link.aps.org/doi/10.1103/PhysRevLett.100.220502.

[132] S. Wiesner, *Conjugate coding*, SIGACT News, 15 (1983), p. 78–88, https://doi.org/10.1145/1008908.1008920, https://doi.org/10.1145/1008908.1008920.

[133] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim, *Benchmarking an 11-qubit quantum computer*, Nature Communications, 10 (2019), p. 5464, https://doi.org/10.1038/s41467-019-13534-2, https://doi.org/10.1038/s41467-019-13534-2.

[134] Q. Xu, X. Tan, and R. Huang, *Improved resource state for verifiable blind quantum computation*, Entropy, 22 (2020), https://doi.org/10.3390/e22090996, https://www.mdpi.com/1099-4300/22/9/996.

[135] A. C.-C. Yao, *How to generate and exchange secrets*, in Proceedings of the 27th Annual Symposium on Foundations of Computer Science, SFCS '86, USA, 1986, IEEE Computer Society, p. 162–167, https://doi.org/10.1109/SFCS.1986.25, https://doi.org/10.1109/SFCS.1986.25.

[136] J. Zou, Z. Wei, S. Sun, X. Liu, and W. Wu, *Quantum circuit implementations of aes with fewer qubits*, in Advances in Cryptology – ASIACRYPT 2020, S. Mo-

riai and H. Wang, eds., Cham, 2020, Springer International Publishing, pp. 697–726, https://doi.org/10.1007/978-3-030-64834-3_24, https://link.springer.com/chapter/10.1007%2F978-3-030-64834-3_24.