Unifying Quantum Verification and Error-Detection: Theory and Tools for Optimisations

Theodoros Kapourniotis[1], Elham Kashefi[2,3], Dominik Leichtle[2], Luka Music[4], and Harold Ollivier[5]

[1] Department of Physics, University of Warwick, Coventry CV4 7AL, United Kingdom
[2] Laboratoire d'Informatique de Sorbonne Université, CNRS, Sorbonne Université, 75005 Paris, France
[3] School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, United Kingdom
[4] Quandela, 91300 Massy, France
[5] INRIA, 75012 Paris, France

Abstract. With the recent availability of cloud quantum computing services, the question of verifying quantum computations delegated by a client to a quantum server is becoming of practical interest. While Verifiable Blind Quantum Computing (VBQC) has emerged as one of the key approaches to address this challenge, current protocols still need to be optimised before they are truly practical. To this end, we establish a fundamental correspondence between error-detection and verification and provide sufficient conditions to both achieve security in the Abstract Cryptography framework and optimise resource overheads of all known VBQC-based protocols. As a direct application, we demonstrate how to systematise the search for new efficient and robust verification protocols for BQP computations. While we have chosen Measurement-Based Quantum Computing (MBQC) as the working model for the presentation of our results, one could expand the domain of applicability of our framework via direct known translation between the circuit model and MBQC.

1 Introduction

1.1 Context

Secure delegation of quantum computation is a long-standing topic of research where a client wants to perform a computation on a remote server, without necessarily trusting it. In this context, a computation is deemed blind when the privacy of the data and algorithm is guaranteed, and verified whenever the integrity of the computation is guaranteed or else the computation has aborted. None of these criteria are specific to quantum computing as users have always needed to protect their data, their algorithmic know-how and ensure that no party can manipulate results beyond their ability to choose their inputs [13,12]. Initially, the main interest for verifying quantum computations was relative to the nature of the client (or verifier) [1,27]: what quantum power is needed by the client to verify a possibly unbounded quantum server (or prover)? Yet, this topic has gained attention due to the recent development of remotely accessible quantum computers, where no cryptographic guarantee is currently provided to clients delegating their computations to service providers. This, in turn, transformed a mostly theoretical question into a more practical one.

The first line of work to tackle this question introduced protocols guaranteeing statistical security by requiring the client to perform single qubit operations – either preparations or measurements. More recent protocols provide only computational security, with the benefit of being applicable to fully classical clients. In addition to the theoretical implications raised by verification, the possible practicality of proposed protocols has always been an important aspect of research on this topic as it was anticipated that quantum computers would be mostly available remotely. Recent years have confirmed this direction. Existing end-users of quantum computing services often emphasise the importance of integrity guarantees for the computations they delegate, as well as privacy of their data and algorithms.

Several protocols have been introduced along the years with the purpose of lowering some of the resource overhead of secure delegated computations. Yet, there is a lack of theoretical understanding of the requirements to construct robust and efficient verification protocols, as well as a lack of tools to systematise their optimisation. More precisely, while there are protocols that optimise the qubit communication, the complexity of the operations on the client's side, the overhead on the server's side, or the amount of tolerable noise, none provide general methods that could be applied when designing the protocols themselves and used to tailor their performance to specific contexts and use-cases.

In this work, we deconstruct composable and statistically secure protocols for delegated quantum computations, to both exhibit their fundamental structure and allow for their convenient optimisation. We focus on protocols framed in the Measurement Based Quantum Computation (MBQC) model [26]. Our results are based on the simple yet powerful ideas that *detecting deviations from the client's instructions which are potentially harmful for the computation should yield verification*, while *the ability to be insensitive to those that are not harmful should provide noise-robustness*. We formalise this intuition through the concept of trappified schemes – a set of computations containing factitious computations whose results are known only to the client –, together with a necessary condition for obtaining negligible security errors with polynomial resources. Even more importantly, this work naturally connects the field of error-detection to that of verification, opening considerably the sources of inspiration for designing new trappified schemes and thus verification protocols.

As a concrete application, we construct a generic compiler for verifying BQP computations without any overhead of physical resources compared to the unprotected computation. Its efficiency is then optimised thanks to the introduction of new traps inspired by syndrome measurements of error-correcting codes.

Related Work. The first verification protocols have relied on the client's ability to access a small constant size quantum machine. It serves to encrypt the instructions delegated to the server or to perform the necessary operations to complete the computation once a complex resource state is provided by the server [2,3,10,4,15]. In both cases, the behaviour of the server is checked thanks to insertion of smaller computations alongside the one of interest whose result is known to the client.

More recent protocols used the mapping of BQP computations onto the 2-local Hamiltonian problem. In [11,14], the necessity of encryption was removed while the client was still required to perform X and Z measurements. In the ground breaking work of [21], the client was made entirely classical at the expense of some post-quantum secure computational assumptions.

Unfortunately, all these protocols – even those with a classical client – are too resource-intensive on the server's side to be practical. Several efforts have been devoted to improve the situation, in particular for protocols using Universal Blind Quantum Computing to encrypt the instructions sent to the server. [19,29] seek to reduce the connectivity of the graph supporting the computation; [18] reduces the communication instead ; and the objective of [9] is to limit further the set of operations that the client must wield to be able to perform the protocol. Recently, [20] considers the joint optimisation of the space overhead as well as the level of honest noise that the protocol is able to withstand while still accepting.

1.2 Overview of results

In this paper, we express our results in the prepare-and-send model trading generality for simplicity, whereas we rely on the equivalence with the receive-and-measure model to extend their applicability [28]. In this model, the client is prepares a small subset of quantum states, performs limited single-qubit operations and sends its prepared states to a server via a quantum communication channel. The server then executes the client's instructions and possibly returns some quantum output via the same quantum channel. As we seek not only verification but also blindness, we will use extensively the simple obfuscation technique put forth in the Universal Blind Quantum Computation (UBQC) protocol (see Section 2 for basics about UBQC) and consisting in randomly rotating each individual qubit sent by the client to the server.

The main idea that has been put at work in previous verification protocols is that, in such case, the client can chose to insert some factitious computations alongside the one it really intends to delegate. Because the client can choose factitious computations whose results are easy to compute classically and therefore to test, and because the server does not know whether the computation is genuine or factitious, these allows to ensure that the server is non-malicious.

Analyzing Deviations with Traps (Section 3). Here, we lay out a series of concepts that formally define theses factitious computations, or traps, as probabilistic error-detecting schemes. More precisely, we define *trappified canvases* as sub-computations on an MBQC graph with a fixed input state and classical outputs which follow a probability distribution that is efficiently computable classically.

This is paired to a decision function which, depending on the output of this sub-computation, returns whether the trap accepts or rejects. The term canvas refers to the fact that there is still empty space on the graph alongside the factitious computation for the client's computation to be "painted into". This task is left to an *embedding algorithm*, which takes a computation and a trappified canvas and fills in the missing parts so that the output is a computation containing both the client's computation and a trap.

Because we aim at blind delegating the execution of trappified canvases to a possibly fully malicious server that can deviate adaptively, a single trappified canvas will not be enough to constraint its behaviour significantly. Instead we randomise the construction of trappified canvases, and in particular the physical location of the trap. This gives rise to the concept of *trappified schemes* (Definition 8) which are sets of trappified canvases from which the client can sample efficiently.

Additionally, for these constructs to be useful in blind protocols they need to satisfy two properties. First, no information should leak to the server when it is using one trappified canvas over another. This means that executing one trappified canvas or another must be indistinguishable to the server. If this is the case, we say that they are *blind-compatible*. Second, no information should leak to the server about the computation in spite of being embedded into a larger computation that contains a trap. This implies that the decision to accept or reject the computation should not be depending on the client's desired computation. If this is the case, we call the embedding a *proper embedding*.

Finally, we examine the effect of deviations on individual trappified canvases as well as on trappified schemes. More precisely, we categorise deviations with the help of trappified schemes as follows: (i) if the scheme rejects with probability $(1 - \epsilon)$, then it ϵ-detects the deviation; (ii) if the scheme accepts with probability $1 - \delta$, it is δ-insensitive to the deviation; and finally (iii) if the result of all possible computations of interest is correct with probability $1 - \nu$, then the scheme is ν-correct for this deviation.

Secure Verification from Trap Based Protocols (Section 4). Here, our contribution is a series of theorems that give general design rules for constructing secure, efficient and robust verification protocols based on the detection, insensitivity and correctness properties of trappified schemes.

We start by constructing a natural Prepare-and-send protocol from any trappified scheme, see the informal Protocol 1.

Protocol 1 Trappified Delegated Blind Computation Protocol (Informal)

1. The Client samples a trappified canvas from the trappified scheme and embeds its computation, yielding a trappified pattern.
2. The Client blindly delegates this trappified pattern to the Server using the UBQC Protocol, after which the Client obtains the output of the trappified pattern.
3. The Client decides whether to abort or not based on the result of the decision function of the trappified canvas.
4. If it didn't abort, the Client performs some simple classical or quantum post-processing on the output.

We then address the following question: what are the conditions required for these error-detection mechanisms to provide verification? The following theorem states that the trappified scheme should detect with high probability all errors for which the computation is not correct.

Theorem 1 (Detection Implies Verifiability, Informal). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two sets of Pauli deviations such that $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$, and $\mathsf{I} \in \mathcal{E}_2$. If the Trappified Delegated Blind Computation Protocol uses a trappified scheme which ϵ-detects $\mathcal{E}_1$, is δ-insensitive to $\mathcal{E}_2$, is ν-correct on $\mathcal{G}_V \setminus \mathcal{E}_1$, then the protocol is $\max(\epsilon, \delta + \nu)$-secure.*

In other words, it is acceptable to not detect a deviation so long as it has only little effect on the result of the computation of interest. This intuitive result is proved in the framework of Abstract Cryptography [23]. We introduce novel techniques to derive the protocol's composable security directly, without resorting to local criteria as in [8]. We construct a simulator that is able to correctly guess whether to accept or reject its interaction with the server without ever knowing what the client's computation is, thereby reproducing the behaviour of the concrete protocol although it is accessing a secure-by-design ideal delegated quantum computation resource. As such, this provides the first direct proof of composable security of the original VBQC protocol [10].

We next examine the conditions under which the protocol is robust against honest noise. We show that it is sufficient for the trappified scheme to be both insensitive to and correct on likely errors generated by the noise model.

Theorem 2 (Robust Detection Implies Robust Verifiability, Informal). *We assume that the server in the Trappified Delegated Blind Computation Protocol is honest-but-noisy: the error applied is in $\mathcal{E}_2$ with probability $(1-p_2)$ and $\mathcal{G}_V \setminus \mathcal{E}_2$ with probability p_2. Then, the client accepts with probability at least $(1-p_2)(1-\delta)$, and if accepted the distance between the implemented transformation and the client's computation is bounded by $\nu + p_2 + \delta$.*

We conclude this theoretical deconstruction of verification protocols by exploring the necessary conditions for obtaining a security error which is exponentially close to zero without blowing up the server's memory requirements. We show that efficient trappified schemes must incorporate some error-correction mechanism.

Theorem 3 (Error-Correction Prevents Resource Blow-up, Informal). *Assume that the Trappified Delegated Blind Computation Protocol has a negligible security error with respect to a security parameter λ. If the size of the output in the trappified pattern is the same as an unprotected execution of the Client's computation for a non-negligible fraction of trappified canvases in the trappified scheme used in the protocol, then the size of the common graph state required to implement the trappified patterns scales super-polynomially in λ.*

These results reveal the strong interplay between the deviation detection properties of trappified schemes and the properties of the corresponding prepare-and-send verification protocol. As a consequence, the optimisation of verification protocols translates into tailoring the deviation detection properties of trappified schemes to specific needs, for which the rich tools of error-correction can be used. This is the focus of the rest of the paper.

Correctness and Security Amplification for Classical Input-Output Computations (Section 5). Here, we construct a general compiler for obtaining trappified schemes. It interleaves separate computations and test rounds in a way inspired by [20]. As a consequence, the overhead for protocols based on such schemes is simply a repetition of operations of the same size as the client's original computation, meaning that verification comes for free so long as the client and server can run the blind protocol. Using our correspondence between error-detection and verification, we then show that this compiler's parameters can be chosen to boost the constant detection and insensitivity rates of the individual test rounds to exponential levels after compilation.

Theorem 4 (From Constant to Exponential Detection and Insensitivity Rates, Informal). *Let $\boldsymbol{P}$ be a trappified scheme and $\boldsymbol{P}'$ be the compiled version described above for n rounds combining a number of tests and computations which are both linear in n. If $\boldsymbol{P}$ ϵ-detects error set $\mathcal{E}_1$ and is δ-insensitive to $\mathcal{E}_2$, then there exists k_1, k_2 linear in n and ϵ', δ' exponentially-low in n such that $\boldsymbol{P}'$ ϵ'-detects errors with more than k_1 errors on all rounds from set $\mathcal{E}_1$ and is δ-insensitive to errors with less than k_2 errors from set $\mathcal{E}_2$.*

This however not enough to obtain negligible security and, as per Theorem 3, we must recombine the results of the computation rounds to correct for these low-weight errors which are not detected. This is done by using a simple majority vote on the computation round outcomes, so that correctness can be independently amplified to an exponential level by using polynomially many computation rounds.

Theorem 5 (Exponential Correctness from Majority Vote, Informal). *There exists k linear in n and ν exponentially-low in n such that $\boldsymbol{P}'$ is ν-correct so long as there are no more than k errors.*

In doing so, we have effectively untangled what drives correctness, security and robustness, thereby considerably simplifying the task of designing and optimising new protocols. More precisely, we can now focus only on the design of the test rounds as their performance greatly influences the value of exponents in the exponentials from the two previous theorems.

New Optimised Trappified Schemes from Stabiliser Testing (Section 6). In this section, we design test rounds and characterise their error-detection and insensitivity properties. This allows to recover the standard traps used in several other protocols, while also uncovering new traps that correspond to syndrome measurements of stabiliser generators – hence once again fruitfully exploiting the correspondence between error-detection and verification.

Finally, we combine all of the above into an optimisation of the deviation detection capability of the obtained trappified schemes that not only beats the current state-of-the-art, but more importantly provides an end-to-end application of our theoretical results.

1.3 Future Work and Open Questions

First, the uncovered connection between error-detection and verification raises further questions such as the extent to which it is possible to infer from the failed traps what the server has been performing.

Second, Theorem 3 implies that some form of error-correction is necessary to obtain exponential correctness. Yet, our protocol shows that sometimes classical error-correction is enough, thereby raising the question of understanding what are the optimal error-correction schemes for given classes of computation that are to be verified.

2 Preliminaries

We introduce here the main components upon which our constructions will rely. Further preliminaries and notations can be found in Appendix A.

The MBQC model of computation emerged from the gate teleportation principle. It was introduced in [26] where it was shown that universal quantum computing can be implemented using graphs-states as resources and adaptive single-qubit measurements. Therefore MBQC and gate-based quantum computations have the same power. The measurement calculus expresses the correspondence between the two models [7]. MBQC works by choosing an appropriate graph state, performing single-qubit measurements on a subset of this state and, depending on the outcomes, apply correction operators to the rest. Quantum computations can be easily delegated in this model by having the client supply the quantum input to the server and instruct it by providing measurement instructions, while the server is tasked with the creation of a large entangled state which is suitable for the client's desired computation.

More precisely, we define the rotation operator around the Z-axis of the Bloch sphere by an angle θ as $\mathsf{Z}(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$ and $|+_\theta\rangle = \mathsf{Z}(\theta)|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$. 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 $\Theta = \left\{\frac{k\pi}{4}\right\}_{k\in\{0,\ldots,7\}}$ [10]. The client's computation is defined by a *measurement pattern* as follows.

Definition 1 (Measurement Pattern). *A* pattern *in the Measurement-Based Quantum Computation model is given by a graph $G = (V, E)$, input and output vertex sets I and O, a flow f which induces a partial ordering of the qubits V, and a set of measurement angles $\{\phi(i)\}_{i\in O^c}$ in the $X - Y$ plane of the Bloch sphere.*

Further details regarding the definition of the flow can be found in references [16,6]. If the client is able to perform single qubit preparations and use quantum communication, it can delegate an MBQC pattern blindly [5], meaning that the Server does not learn anything about the computation besides the prepared graph G, the set of outputs O and the order of measurements. This is done using Protocol 2.

Protocol 2 UBQC Protocol

Client's Inputs: A measurement pattern $(G, I, O, \{\phi(i)\}_{i\in O^c}, f)$ and a quantum register containing the input state ρ_C on qubits $i \in I$.

Protocol:

1. The Client sends the graph's description (G, I, O) and the measurement order to the Server;
2. The Client prepares and sends all the qubits in V to the Server:[6]
   (a) For $i \in I$, it chooses a random bit $a(i)$. For $i \in I^c$, it sets $a(i) = 0$.
   (b) For $i \in O$, it chooses a random bit $r(i)$ and sets $\theta(i) = (r(v) + a_N(v))\pi$ where $a_N(i) = \sum_{j\in N_G(i)} a(j)$. For $i \in O^c$, it samples a random $\theta(i) \in \Theta$.
   (c) For $i \in I$, it sends $\bigotimes_{i\in I} \mathsf{Z}_i(\theta(i))\mathsf{X}_i^{a(i)}(\rho_C)$. For $i \in I^c$ it sends $\left|+_{\theta(i)}\right\rangle$.
3. The Server applies a CZ gate between qubits i and j if (i, j) is an edge of G;
4. For all $i \in O^c$, in the order specified by the flow f, the Client computes the measurement angle $\delta(i)$ and sends it to the Server, receiving in return the corresponding measurement outcome $b(i)$:

$$s_X(i) = \bigoplus_{j\in S_X(i)} b(i) \oplus r(i),\ s_Z(i) = \bigoplus_{j\in S_Z(i)} b(i) \oplus r(i), \tag{1}$$

$$\delta(i) = (-1)^{a(i)}\phi'(i) + \theta(i) + (r(i) + a_N(i))\pi, \tag{2}$$

   where $\phi'(i)$ is computed using Equation 13 with the new values of $s_X(i)$ and $s_Z(i)$;
5. The Server sends back the output qubits $i \in O$;
6. The Client applies $\mathsf{Z}_i^{s_Z(i)+r(i)}\mathsf{X}_i^{s_X(i)+a(i)}$ to the received qubits $i \in O$.

Note that if the output of the client's computation is classical, the set O is empty and the client only receives measurement outcomes. The output measurement outcomes $b(i)$ sent by the Server need to be decrypted by the Client according to the equation $s(j) = b(j) \oplus r(j)$, thus preserving the confidentiality of the output of the computation.

The security of our protocols will be expressed in the Abstract Cryptography framework, recalled in Appendix A.3. The ideal Resource 1 captures the security properties of a blind and verifiable delegated protocol for a given class of computations. It allows a single Client to run a quantum computation on a Server so that the Server cannot corrupt the computation and does not learn anything besides a given leakage l_ρ. We recall the original definition from [8, Definition 4.2].

[6] In the original UBQC Protocol from [5], the outputs are prepared by the Server in the $|+\rangle$ state and are encrypted by the computation flow. In the verification protocol in which we will use the UBQC Protocol later, some inputs to auxiliary trap computations may be included in the global output, meaning that all output qubits must also be prepared by the Client. This does not change the security properties of the UBQC Protocol.

Resource 1 Secure Delegated Quantum Computation

Public Information: Nature of the leakage l_{ρ_C}.

Inputs:

– The Client inputs the classical description of a computation C from subspace $\Pi_{I,C}$ to subspace $\Pi_{O,C}$ and a quantum state ρ_C in $\Pi_{I,C}$.

– 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:

1. If $e = 1$, the Resource sends the leakage l_ρ to the Server's interface; if it receives $c = 1$, the Resource outputs $|\bot\rangle\langle\bot| \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|$ at the Client's output interface.
2. Otherwise it outputs $\mathsf{C}(\rho_C) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|$ at the Client's output interface.

3 Analysing Deviations with Traps

The goal of this section is to introduce the concepts and tools for detecting deviations from a given computation. Later, in Section 4, we combine these techniques with blindness in order to detect malicious deviations, i.e. perform verification. Examples for the constructions below can be found in Appendix B.

3.1 Abstract Definitions of Traps

We start by defining partial MBQC patterns in Definition 2, which fix only a subset of the measurement angles and flow conditions on a given graph. We constrain the flow such that the determinism of the computation is preserved on the partial pattern independently of the how the rest of the flow is specified.

Definition 2 (Partial MBQC Pattern). *Given a graph $G = (V, E)$, a partial pattern P on G is defined by: (i) $G_P = (V_P, E_P = E \cap V_P \times V_P)$, a subgraph of G; (ii) I_P and O_P, the partial input and output vertices, with subspaces $\Pi_{I,P}$ and $\Pi_{O,P}$ defined on vertices I_P and O_P through bases $\mathcal{B}_{I,P}$ and $\mathcal{B}_{O,P}$ respectively; (iii) $\{\phi(i)\}_{i \in V_P \setminus O_P}$, a set of measurement angles; (iv) $f_p : V_P \setminus O_P \to V_P \setminus I_P$, a flow inducing a partial order $\preceq_P$ on V_P.*

We now use this notion to define trappified canvases. These contain a partial pattern whose input state is fixed such that it produces a sample from an easy to compute probability distribution when its ouput qubits are measured in the X basis. These partial patterns are called *traps* and will be used to detect deviations in the following way. Whenever a trap computation is executed, it should provide outcomes that are compatible with the trap's probability distribution. Failure to do so is a sign that the server deviated from the instructions given by the client.

Definition 3 (Trappified Canvas). *A* trappified canvas *$(T, \sigma, \mathcal{T}, \tau)$ on a graph $G = (V, E)$ consists of (i) T, a partial pattern on a subset of vertices V_T of G with input and output sets I_T and O_T; (ii) σ, a tensor product of single-qubit states on $\Pi_{I,T}$; (iii) $\mathcal{T}$, an efficiently classically computable probability distribution over binary strings; (iv) and τ, an efficient classical algorithm that takes as input a sample from $\mathcal{T}$ and outputs a single bit; such that the X-measurement outcomes* *of qubits in O_T are drawn from probability distribution $\mathcal{T}$. Let t be such a sample, the outcome of the trappified canvas is given by $\tau(t)$. By convention we say that it accepts whenever $\tau(t) = 0$ and rejects for $\tau(t) = 1$.*

We will often abuse the notation and refer to the trappified canvas $(T, \sigma, \mathcal{T}, \tau)$ as T. Note that the input and output qubits of a partial pattern may not be included in the input and output qubits of the larger MBQC graph. This gives us more flexibility in defining trappified canvases: during the protocol presented in the next section, the server will measure all qubits in O^c – which may include some of the trap outputs –, while any measurement of qubits in O will be performed by the client. This allows the trap to catch deviations on the output qubits as well.

In order to be useful, trappified canvases must contain enough empty space – vertices which have been left unspecified – to accommodate the client's desired computation. Inserting this computation is done via an *embedding algorithm* as described in the following Definition.

Definition 4 (Embedding Algorithm). *Let $\mathfrak{C}$ be a class of quantum computations. An* embedding algorithm *$E_{\mathfrak{C}}$ for $\mathfrak{C}$ is an efficient classical probabilistic algorithm that takes as input (i) $\mathsf{C} \in \mathfrak{C}$, the computation to be embedded; (ii) $G = (V, E)$, a graph, and an output set O; (iii) T, a trappified canvas on graph G; (iv) $\preceq_G$, a partial order on V which is compatible with the partial order defined by T.*

It outputs (i) a partial pattern C on $V \setminus V_T$, with input and output vertices $I_C \subset V \setminus V_T$ and $O_C = O \setminus O_T$; (ii) two subspaces (resp.) $\Pi_{I,C}$ and $\Pi_{O,C}$ of (resp.) I_C and O_C with bases (resp.) $\mathcal{B}_{I,C}$ and $\mathcal{B}_{O,C}$; (iii) a decoding algorithm $\mathsf{D}_{O,C}$; such that the flow f_C of partial pattern C induces a partial order which is compatible with $\preceq_G$. If $E_{\mathfrak{C}}$ is incapable of performing the embedding, it outputs $\perp$.

As will be come apparent in later definitions, a good embedding algorithm will yield patterns which apply a desired computation C to any input state in subspace $\Pi_{I,C}$, with the output being in subspace $\Pi_{O,C}$ after the decoding algorithm has been run. The decoding algorithm can be quantum or classical depending on the nature of the output. We will furthermore require all embedding algorithms in the paper to have the following property.

Definition 5 (Proper Embedding). *We say that an embedding algorithm $E_{\mathfrak{C}}$ is* proper *if, for any computation $\mathsf{C} \in \mathfrak{C}$ and trappified canvas T that do not result in a $\perp$ output, we have that (i) f_C does not induce dependencies on vertices V_T of partial pattern T and (ii) the input and output subspaces $\Pi_{I,C}$ and $\Pi_{O,C}$ do not depend on the trappified canvas T.*

Definition 6 (Trappified Pattern). *Let $E_{\mathfrak{C}}$ be an embedding algorithm for $\mathfrak{C}$. Given a computation $\mathsf{C} \in \mathfrak{C}$ and a trappified pattern T on graph G with order $\preceq_G$, we call the completed pattern $C \cup T$ which is the first output of $E_{\mathfrak{C}}(\mathsf{C}, G, T, \preceq_G)$ a* trappified pattern.

While embedding a computation in a graph that has enough space for it might seem simple, the hard part is to ensure that the embedding is *proper*. This property implies that no information is carried via the flow of the global pattern from the computation to the trap and it is essential for the security of the verification protocol built using trappified canvases. This can be done either by breaking the graph using the states initialised in $|0\rangle$ or by separating runs for tests and computations. Satisfying this condition using other methods is left as an open question.

Note that the input and output qubits of the computation C might be constrained to be in (potentially strict) subspaces $\Pi_{I,C}$ and $\Pi_{O,C}$ of I_C and O_C respectively. This allows for error-protected inputs and outputs, without having to specify any implementation for the error-correction scheme. In particular, it encompasses encoding classical output data as several, possibly noisy, repetitions which will be decoded by the client through a majority vote as introduced in [20]. It also allows to take into account the case where the trappified pattern comprises a fully fault-tolerant MBQC computation scheme for computing C using topological codes as described in [25].

For verification, our scheme must be able to cope with malicious behaviour: detecting deviations is useful for verification only so long as the server cannot adapt its behaviour to the traps that it executes. Otherwise, it could simply decide to deviate exclusively on non-trap qubits. This is achieved by executing the patterns in a blind way so that the server has provably no information about the location of the traps and cannot avoid them with high probability. To this end, we define *blind-compatible* patterns as those which share the same graph, output vertices and measurement order of their qubits. The UBQC Protocol described in Appendix 2 leaks exactly this information to the server, meaning that it cannot distinguish the executions of two different blind-compatible patterns.

Definition 7 (Blind-Compatibility). *A set of patterns $\boldsymbol{P}$ is* blind-compatible *if all patterns $P \in \boldsymbol{P}$ share the same graph G, the same output set O and there exists a partial ordering $\preceq_{\boldsymbol{P}}$ of the vertices of G which is an extension of the partial ordering defined by the flow of any $P \in \boldsymbol{P}$. This definition can be extended to a set of trappified canvases $\boldsymbol{P} = \{(T, \sigma, \mathcal{T}, \tau)\}$. The partial order $\preceq_{\boldsymbol{P}}$ is required to be an extension of the orderings $\preceq_T$ of partial patterns T.*

A single trap is usually not sufficient to catch deviations on more than a subset of positions of the graph. In order to catch all deviations, it is then necessary to randomise the blind delegated execution over multiple patterns. We therefore define a trappified scheme as a set of blind-compatible trappified canvases which can be efficiently sampled according to a given distribution, along with an algorithm for embedding computations from a given class into all the canvases.

Definition 8 (Trappified Scheme). *A* trappified scheme $(\boldsymbol{P}, \preceq_G, \mathcal{P}, E_{\mathfrak{C}})$ *over a graph G for computation class $\mathfrak{C}$ consists of (i) $\boldsymbol{P}$, a set of blind-compatible trappified canvases over graph G with common partial order $\preceq_{\boldsymbol{P}}$; (ii) $\preceq_G$, a partial ordering of vertices V of G that is an extension of $\preceq_{\boldsymbol{P}}$; (iii) $\mathcal{P}$, a probability distribution over the set $\boldsymbol{P}$ which can be sampled efficiently; (iv) $E_{\mathfrak{C}}$, an proper* *embedding algorithm for $\mathfrak{C}$; such that for all $\mathsf{C} \in \mathfrak{C}$ and all trappified canvases $T \in \boldsymbol{P}$, $E_{\mathfrak{C}}(\mathsf{C}, G, T, \preceq_G) \neq \perp$, i.e. any computation can be embedded in any trappified canvas using the common order $\preceq_G$.*

Without loss of generality, in the following, the probability distribution used to sample the trappified canvases will generally be $u(\boldsymbol{P})$, the uniform distribution over $\boldsymbol{P}$. The general case can be approximated from the uniform one with arbitrary fixed precision by having several copies of the same canvas in $\boldsymbol{P}$. We take $T \sim \boldsymbol{P}$ to mean that the trappified canvas is sampled according to the distribution $\mathcal{P}$ of trappified scheme $\boldsymbol{P}$.

Note that in Definition 8 above, while the blindness condition ensures that a completed patterns obtained after running the embedding algorithm hides the location of the traps, the existence of a partial order $\preceq_G$ compatible with that of the trappified canvases ensures that this remains true when considering the scheme as a whole, i.e the order in which the qubits are measured does not reveal information about the chosen trappified canvas itself, which would otherwise break the blindness of the scheme.

3.2 Effect of Deviations on Traps

We can now describe the purpose of the objects described in the previous subsection, namely detecting the server's deviations from their prescribed operations during a given delegated computation. We start by recalling that the blindness of UBQC Protocol is obtained by Pauli-twirling the operations delegated to the server. This implies that any deviation can be reduced to a convex combination of Pauli operators. Then, we formally define Pauli deviation detection and insensitivity for trappified canvases and schemes. We show in the next section that these key properties are sufficient for obtaining a verifiable delegated computation by formalising the steps sketched here.

When a client delegates the execution of a pattern P to a server using Protocol 2, the server can potentially deviate in an arbitrary way from the instructions it receives. By converting into quantum states both the classical instructions sent by the client – i.e. the measurement angles – and the measurement outcomes sent back by the server, all operations on the server's side can be modelled as a unitary F acting on all the qubits sent by the client and some ancillary states $|0\rangle_S$, before performing measurements in the computational basis to send back the outcomes $|b\rangle$ that the client expects from the server.

The instructions of the server in an honest execution of the UBQC Protocol 2 correspond to:

1. entangling the received qubits corresponding to the vertices of the computation graph with operation $E_G = \bigotimes_{(i,j) \in E} \mathsf{CZ}_{i,j}$;
2. performing rotations on non-output vertices around the Z-axis, controlled by the qubits which encode the measurement angles instructed by the client;
3. applying a Hadamard gate H on all non-output vertices;
4. measuring non-output vertices in the $\{|0\rangle, |1\rangle\}$ basis.

The steps (i-iii) correspond to a unitary transformation U_P that depends only on the public information that the server has about the pattern P – essentially the computation graph G and an order of its vertices compatible with the flow of P. Hence, the unitary part U_P of the honestly executed protocol for delegating P can always be extracted from F, so that $\mathsf{F} = \mathsf{F}' \circ \mathsf{U}_P$. Here, F' is called a *pure deviation* and is applied right before performing the computational basis measurements for non-output qubits and right before returning the output qubits to the Client. When the pattern is executed blindly using Protocol 2, the state in the server's registers during the execution is a mixed state over all possible secret parameters chosen by the client. It is shown in [17] that the resulting summation over the secret parameters which hide the inputs, measurement angles and measurement outcomes is equivalent to applying a Pauli twirl to the pure deviation F'. This effectively transforms it into a convex combination of Pauli operations applied after U_P.

Hence, any deviation by the server can be represented without loss of generality by choosing with probability $\Pr[\mathsf{E}]$ an operator E in the Pauli group $\mathcal{G}_V$ over the vertices V of the graph used to define P, and executing $\mathsf{E} \circ \mathsf{U}_P$ instead of U_P for the unitary part of the protocol. By a slight abuse of notation, such transformation will be denoted $\mathsf{E} \circ P$. Furthermore, if $C \cup T$ is a trappified pattern obtained from a trappified canvas T that samples $t = (t_1, \ldots, t_N)$ from the distribution $\mathcal{T}$, then in the presence of deviation E, it will sample from a different distribution. For instance, whenever E applies a Z operator on a vertex, it can be viewed as an execution of a pattern where the angle δ for this vertex is changed into $\delta + \pi$. Whenever E applies a X operator on a vertex, δ is transformed into $-\delta$. We now give a lemma which will be useful throughout the rest of the paper.

Lemma 1 (Independence of Trap and Computation). *Let $C \cup T$ be a trappified pattern obtained from the trappified canvas T which samples from distribution $\mathcal{T}$ through a proper embedding algorithm of computation C. Then, for all Pauli errors E, the distribution of trap measurement outcomes is independent of the computation C and of the input state in the subspace $\Pi_{I,C}$.*

Proof. Let f_C be the flow of computation of the embedded computation C. Because the embedding is proper according to Definition 4, the dependencies induced by f_C do not affect trap qubits V_T. Furthermore, the input of the trap is fixed along with its partial pattern, independently of the computation. Therefore, the distribution of the trap measurement outcomes is also independent of the embedded computation being performed on the rest of the graph as well as the input state of such computation. ■

Indeed, for a completed trappified pattern $C \cup T$ obtained by embedding a computation C onto a trappified canvas T, the action of E on the vertices outside V_T does not have an impact on the measurement outcomes of the vertices in V_T. This allows to define the trap outcome distribution under the influence of error E solely as a fuction of E and $\mathcal{T}$. Such modified distribution is denoted $\mathsf{E} \circ \mathcal{T}$.

As an additional consequence, it is possible to define what it means for a given trappified canvas to detect and to be insensitive to Pauli errors:

Definition 9 (Pauli Detection). *Let T be a trappified canvas sampling from distribution $\mathcal{T}$. Let $\mathcal{E}$ be a subset of $\mathcal{G}_V$. For $\epsilon > 0$, we say that T* ϵ-detects $\mathcal{E}$ *if for all $\mathsf{E} \in \mathcal{E}$ we have $\Pr_{t \sim \mathsf{E} \circ \mathcal{T}}[\tau(t) = 1] \geq 1 - \epsilon$. We say that a trappified scheme $\boldsymbol{P}$* ϵ-detects $\mathcal{E}$ *if for all $\mathsf{E} \in \mathcal{E}$ we have $\sum_{T \in \boldsymbol{P}} \Pr_{\substack{T \sim \mathcal{P} \\ t \sim \mathsf{E} \circ \mathcal{T}}}[\tau(t) = 1, T] \geq 1 - \epsilon$.*

Definition 10 (Pauli Insensitivity). *Let T be a trappified canvas sampling from distribution $\mathcal{T}$. Let $\mathcal{E}$ be a subset of $\mathcal{G}_V$. For $\delta > 0$, we say that T is* δ-insensitive to $\mathcal{E}$ *if for all $\mathsf{E} \in \mathcal{E}$ we have $\Pr_{t \sim \mathsf{E} \circ \mathcal{T}}[\tau(t) = 0] \geq 1 - \delta$. We say that a trappified scheme $\boldsymbol{P}$ is* δ-insensitive to $\mathcal{E}$ *if for all $\mathsf{E} \in \mathcal{E}$ we have $\sum_{T \in \boldsymbol{P}} \Pr_{\substack{T \sim \mathcal{P} \\ t \sim \mathsf{E} \circ \mathcal{T}}}[\tau(t) = 0, T] \geq 1 - \delta$.*

Above, the probability distribution stems both from the randomness of quantum measurements of the trap output qubits yielding the bit string t, and the potentially probabilistic nature of the decision function τ. In the case of trappified schemes, the probability distribution for obtaining a given result for τ also depends on the choice of canvas $T \in \boldsymbol{P}$, sampled according to the probability distribution $\mathcal{P}$.

In the same spirit, there are physical deviations that nonetheless produce little effect on the computations embedded into trappified canvases and trappified schemes. When they occur, the computation is still almost correct.

Definition 11 (Pauli Correctness). *Let $(T, \sigma, \mathcal{T}, \tau)$ be a trappified canvas and $E_{\mathfrak{C}}$ an embedding algorithm. Let $C \cup T$ be the pattern obtained by embedding a computation $\mathsf{C} \in \mathfrak{C}$ on T using $E_{\mathfrak{C}}$ and let $|\psi\rangle$ be a state in $I_C \otimes R$, for sufficiently large auxiliary system R, such that $\mathrm{Tr}_R(|\psi\rangle) \in \Pi_{I,C}$, where $\Pi_{I,C}$ is the client's input subspace. Let $\mathcal{E}$ be a subset of $\mathcal{G}_V$. For $\mathsf{E} \in \mathcal{E}$, we define $\tilde{\mathsf{C}}_{T,\mathsf{E}} = \mathsf{D}_{O,C} \circ \mathrm{Tr}_{O_C^c} \circ \mathsf{E} \circ (C \cup T)$ to be the CPTP map resulting from applying the trappified pattern $C \cup T$ followed by the decoding algorithm $\mathsf{D}_{O,C}$ on the output of the computation. For $\nu \geq 0$, we say that T is* ν-correct *on $\mathcal{E}$ if:*[7]

$$\forall \mathsf{E} \in \mathcal{E},\ \forall \mathsf{C} \in \mathfrak{C},\ \max_{\psi} \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C} \otimes \mathsf{I}_T) \otimes \mathsf{I}_R(|\psi\rangle\langle\psi| \otimes \sigma)\|_{\mathrm{Tr}} \leq \nu. \tag{3}$$

This is extended to a trappified scheme $\boldsymbol{P}$ by requiring the bound to hold for all $T \in \boldsymbol{P}$.

In the following, sets of deviations that have little effect on the result of the computation according to diamond distance will be called *harmless*, while their complement are *possibly harmful*.

We conclude this section with some remarks regarding basic properties of trappified schemes and a simple but powerful result allowing to construct trappification schemes from simpler ones.

Remark 1 (Existence of Harmless Deviations). Why not just detect all possible deviations rather than count on the possibility that some have little impact on the

[7] Equation 3 corresponds to the diamond norm between the correct and deviated CPTP maps, but with a fixed input subspace and a fixed input for the trap qubits.

actual computation? The reason is that these are plentiful in MBQC. Following our convention to view all measurements as computational basis measurements preceded by an appropriate rotation, any deviation E that acts as I and Z on measured qubits does not change the measurement outcomes and have no effect on the final outcome. Consequently, for classical output computations, only X and Y deviations need to be analysed. These are equivalent to flipping the measurement outcome, which propagate to the output via the flow corrections.

Remark 2 (A Trappified Canvas is a Trappified Scheme). Any trappified canvas T can be seen as a trappified scheme $\boldsymbol{P} = \{T\}$ and the trivial distribution. If the trappified pattern ϵ-detects $\mathcal{E}_1$ and is δ-insensitive to $\mathcal{E}_2$, so is the corresponding trappified scheme.

Remark 3 (Pure Traps). A trappified scheme $\boldsymbol{P}$ may only consist of trappified canvases that cover the whole graph $G = (V, E)$ if $V_T = V$ for all $T \in \boldsymbol{P}$. This corresponds to the special case where the trappified scheme cannot embed any computation, i.e. $\mathfrak{C} = \emptyset$ and the embedding algorithm applied to a canvas $T \in \boldsymbol{P}$ always return T. The detection, insensitivity and correctness properties also apply to this special case, although the correctness is trivially satisfied.

Lemma 2 (Simple Composition of Trappified Schemes). *Let $(\boldsymbol{P}_i)_i$ be a sequence of trappified schemes with corresponding distributions $\mathcal{P}_i$ such that $\boldsymbol{P}_i$ ϵ_i-detects $\mathcal{E}_1^{(i)}$ and is δ_i-insensitive to $\mathcal{E}_2^{(i)}$. Let $(p_i)_i$ be a probability distribution.*

Let $\boldsymbol{P} = \bigcup_i \boldsymbol{P}_i$ be the trappified scheme with the following distribution $\mathcal{P}$:

1. *Sample a trappified scheme $\boldsymbol{P}_j$ from $(\boldsymbol{P}_i)_i$ according to $(p_i)_i$;*
2. *Sample a trappified canvas from $\boldsymbol{P}_j$ according to $\mathcal{P}_j$.*

Let $\mathcal{E}_1 \subseteq \bigcup_i \mathcal{E}_1^{(i)}$ and $\mathcal{E}_2 \subseteq \bigcup_i \mathcal{E}_2^{(i)}$. Then, $\boldsymbol{P}$ ϵ-detects $\mathcal{E}_1$ and is δ-insensitive to $\mathcal{E}_2$ with $1 - \epsilon = \min_{\mathsf{E} \in \mathcal{E}_1} \sum_{i, \mathsf{E} \in \mathcal{E}_1^{(i)}} p_i(1 - \epsilon_i)$, and $1 - \delta = \min_{\mathsf{E} \in \mathcal{E}_2} \sum_{i, \mathsf{E} \in \mathcal{E}_2^{(i)}} p_i(1 - \delta_i)$.

Note that we do not consider above the embedding function. If we assume that all schemes can embed the same set of computations, then it is possible to use the embedding of the one which is chosen at step 1 above. We will see later an example of how to combine trappified schemes with different computation classes in Section 5.

4 Secure Verification from Trap Based Protocols

In this section we use the properties defined above to derive various results which help break down the tasks of designing and proving the security of verification protocols into small and intuitive pieces. We start by giving a description of a general protocol using trappified schemes which encompasses all prepare-and-send MBQC-based protocol aiming to implement the SDQC funtionality (Definition 1). We then relate the security of this protocol in the Abstract Cryptography framework to the ϵ-detection, δ-insensitivity and ν-correctness of the trappified scheme used in the protocol. Consequently, we can from then on only focus on these three properties instead of looking at the full protocol, which already removes a lot of steps in future proofs.

Then we demonstrate how increasing the insensitivity set yields a protocol which is robust to situations where the server is honest-but-noisy with a contained noise parameter. These results further simplify the design of future protocols since many complex proofs can be avoided, allowing us to concentrate on designing more efficient trappified schemes and directly plugging them into the generic protocol and compiler to yield exponentially-secure and noise-robust protocols implementing SDQC. We finally describe a consequence of these results in the case where the security of the protocol is exponentially-low in a given security parameter. We show that this automatically implies that the computation must be protected against low-weight errors if we restrict the server's resources to be polynomial in the security parameter. The proofs of the results from this section can be found in Appendix C.

General Verification Protocol from Trappified Schemes. Given a computation C, it is possible to delegate its trappified execution in a blind way. To do so, the Client simply chooses one trappified canvas from a scheme at random, inserts into it the computation C using an embedding algorithm and blindly delegates the execution of the resulting trappified pattern to the Server. The steps are formally described in Protocol 3.

Protocol 3 Trappified Delegated Blind Computation

Public Information: $\mathfrak{C}$, a class of quantum computations; $G = (V, E)$, a graph with output set O; $\boldsymbol{P}$, a trappified scheme on graph G; $\preceq_G$, a partial order on V compatible with $\boldsymbol{P}$.

Client's Inputs: Computation $\mathsf{C} \in \mathfrak{C}$ and a quantum state ρ_C compatible with C.

Protocol:

1. The Client samples a trappified canvas T from the trappified scheme $\boldsymbol{P}$.
2. The Client runs the embedding algorithm $E_{\mathfrak{C}}$ from $\boldsymbol{P}$ on its computation C, the graph G with output space O, the trappified pattern T, and the partial order $\preceq_G$. It obtains as output the trappified pattern $C \cup T$.
3. The Client and Server blindly execute the trappified pattern $C \cup T$ on input state ρ_C using the UBQC Protocol 2.
4. If the output set is non-empty (if there are quantum outputs), the Server returns the qubits in positions O to the Client.
5. The Client measures the qubits in positions $O \cap V_T$ in the X basis. It obtains the trap sample t.
6. The Client checks the trap by computing $\tau(t)$. It rejects and outputs $(\perp, \mathsf{Rej})$ if $\tau(t) = 1$.
7. Otherwise, the Client accepts the computation. It applies the decoding algorithm $\mathsf{D}_{O,C}$ to the output of Protocol 2 on vertices $O \setminus V_T$ and set the result as its output along with Acc.

Note that this protocol offers blindness not only at the level of the chosen trappified pattern, but also at the level of the trappified scheme itself. More precisely, by delegating the chosen pattern, the client reveals at most the graph of the pattern, a partial order of its vertices and the location of the output qubits of the pattern, if there are any, comprising computation and trap outputs. However, trappified patterns of a trappified scheme are blind-compatible, that is they share the same graph and same set of output qubits. Therefore, the above protocol also hides which trappified pattern has been executed among all possible ones, hence concealing the location of traps.

Blind Deviation Detection Implies Verifiability. We now formalise the following intuitive link between deviation detection and verification in the context of delegated computations. On one hand, if a delegated computation protocol is correct,[8] not detecting any deviation by the server from its prescribed sequence of operations should be enough to guarantee that the final result is correct. Conversely, detecting that some operations have not been performed as specified should be enough for the client to reject potentially incorrect results. Combining those two cases should therefore yield a verified delegated computation.

To this end, we show how the deviation detection capability of trappified schemes is used to perform verification. This is done by proving that Protocol 3 above constructs the Secure Delegated Quantum Computation Resource 1 in the Abstract Cryptography framework. This resource allows a Client to input a computation and a quantum state and to either receive the correct outcome or an abort state depending on the Server's choice, whereas the Server only learns at most some well defined information contained in a leak l_ρ. More precisely, we show that any distinguisher has a bounded distinguishing advantage between the real and ideal scenarios so long as the trappified scheme $\boldsymbol{P}$ detects a large fraction of deviations that are possibly harmful to the computation.

Theorem 6 (Detection Implies Verifiability). *Let $\boldsymbol{P}$ be a trappified scheme with a proper embedding. Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two sets of Pauli deviations such that $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$, and $\mathsf{I} \in \mathcal{E}_2$. If $\boldsymbol{P}$ ϵ-detects $\mathcal{E}_1$, is δ-insensitive to $\mathcal{E}_2$, is ν-correct on $\mathcal{G}_V \setminus \mathcal{E}_1$, for $\epsilon, \delta, \nu > 0$, then the Trappified Delegated Blind Computation Protocol 3 for computing CPTP maps in $\mathfrak{C}$ using $\boldsymbol{P}$ is $\delta+\nu$-correct and $\max(\epsilon, \nu)$-secure in the Abstract Cryptograhy framework, i.e. it $\max(\epsilon, \delta+\nu)$-constructs the Secure Delegated Quantum Computation Resource 1 where the leak is defined as $l_\rho = (\mathfrak{C}, G, \boldsymbol{P}, \preceq_G)$.*

Remark 4 (Using Other Blind Protocols.). In this work we use the UBQC protocol to provide blindness. This protocol is based on the *prepare-and-send* principle. The direct mirror situation, where the Server prepares states and sends them to the Client, is called the *receive-and-measure* paradigm. These are also based on MBQC and were shown to be equivalent to prepare-and-send protocol by [28] using the Abstract Cryptography framework. Our techniques are therefore directly applicable to this setting as well with the same security guarantees. These two setups together cover most protocols that have been designed and which may be implemented in the near future.

The work of [21] introduced an explicit protocol for verifying BQP computations by relying only on classical interactions and a computational hardness

[8] Here we use correctness in a cryptographic setting, meaning that all parties execute as specified their part of the protocol.

assumption. Our techniques are fully applicable as well using a protocol which ϵ_{bl}-*computationally-constructs* the Blind Delegated Quantum Computation Resource 2 in the AC framework and is capable of implementing MBQC computations natively. The resulting protocol is of course computationally-secure only. A simple hybrid argument can be used first to replace any such computationally-secure protocol with Resource 2 first – at a cost of ϵ_{bl} – and then the UBQC protocol at no cost. The other steps of the proof remain unchanged.

Insensitivity Implies Noise-Robustness. Then, we give conditions on protocols implementing SDQC so that they are able to run on noisy machines with a good acceptance probability. We show formally the following intuitive reasoning: if the errors to which the trappified scheme is insensitive do not disturb the computation too much, then a machine which mostly suffers from such errors will almost always lead to the client accepting the computation and the output will be close to perfect.

Theorem 7 (Robust Detection Implies Robust Verifiability). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two sets of Pauli deviations such that $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$ and $\mathsf{I} \in \mathcal{E}_2$. Let $\boldsymbol{P}$ be a trappified scheme for computation set $\mathfrak{C}$, which is δ-insensitive to $\mathcal{E}_2$ and ν-correct on $\mathcal{G}_V \setminus \mathcal{E}_1$. Let $C \cup T$ be a trappified pattern resulting from embedding computation $\mathsf{C} \in \mathfrak{C}$ in trappified canvas T sampled from $\boldsymbol{P}$. We assume an execution of Protocol 3 with an honest-but-noisy Server whose noise is modelled by sampling an error $\mathsf{E} \in \mathcal{E}_2$ with probability $(1-p_2)$ and $\mathsf{E} \in \mathcal{G}_V \setminus \mathcal{E}_2$ with probability p_2. Then, the Client in Protocol 3 accepts with probability at least $(1-p_2)(1-\delta)$, and if accepted the distance between the implemented transformation and the client's computation is bounded as follows:*

$$\forall \mathsf{C} \in \mathfrak{C},\ \max_{\psi} \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C}) \otimes \mathsf{I}_R(|\psi_C\rangle\langle\psi_C| \otimes \sigma)\|_{\mathrm{Tr}} \leq \nu + p_2 + \delta, \tag{4}$$

where $|\psi_C\rangle$ is a purification of the Client's input ρ_C using auxiliary quantum register R, and $\tilde{\mathsf{C}}_{T,\mathsf{E}} = \mathsf{D}_{O,C} \circ \mathrm{Tr}_{O_C^c} \circ \mathsf{E} \circ (C \cup T)$.

This Theorem shows that whenever (i) a noise process generates deviations that are within $\mathcal{E}_2$ with overwhelming probability, (ii) the embedding of the computation C within $\boldsymbol{P}$ adds redundancy in such a way that ν is negligible, and (iii) $\boldsymbol{P}$ is δ-insensitive to $\mathcal{E}_2$ for a negligible δ, then the protocol will accept the computation almost all the time, and the computation will be very close to C. We will see in the next section how these parameters can be amplified. The theorem above shows the importance not only of the parameters of the scheme, but also the size of the sets $\mathcal{E}_1$ and $\mathcal{E}_2$. By creating schemes which have more errors fall in set $\mathcal{E}_2$, it is possible to have a direct impact both in terms of acceptance probability and fidelity in the context of honest-but-noisy executions. We now show that this error-correction is not only necessary for the noise-robustness of the protocol but also its efficiency.

Efficient Verifiability Requires Error-Correction. We now present an important consequence of Theorem 6 in the case where the correctness error $(\delta + \nu)$ and the security error $\max(\epsilon, \nu)$ are negligible with respect to a security parameter λ. We show that this correctness and security regime can only be achieved with a polynomial qubit overhead if the computation is error-protected.

More precisely, we denote $\boldsymbol{P}(\lambda)$ a sequence of trappified schemes indexed by a security parameter λ, such that it $\epsilon(\lambda)$-detects a set $\mathcal{E}_1(\lambda) \subseteq \mathcal{G}_V(\lambda)$ of Pauli deviations, is $\nu(\lambda)$-correct outside $\mathcal{E}_1$, and is $\delta(\lambda)$-insensitive to $\mathcal{E}_2(\lambda) \subseteq \mathcal{G}_V(\lambda) \setminus \mathcal{E}_1(\lambda)$, for $\epsilon(\lambda)$, $\nu(\lambda)$ and $\delta(\lambda)$ negligible in λ. Additionally, let C be a computation pattern which implements the client's desired computation CPTP map $\mathsf{C} \in \mathfrak{C}$ on some input state $|\psi\rangle$. We are now interested in the server's memory overhead introduced by implementing C using $\boldsymbol{P}(\lambda)$ for computation class $\mathfrak{C}$ instead of the unprotected pattern C. This is expressed by the ratio $|G_{\boldsymbol{P}(\lambda)}|/|G_C|$ between the number of vertices in the graph $G_{\boldsymbol{P}(\lambda)}$ common to all canvases in $\boldsymbol{P}(\lambda)$ and the graph G_C used by the pattern C.

For a trappified pattern $C \cup T$ obtained by using the embedding algorithm on a trappified canvas from $\boldsymbol{P}(\lambda)$ we denote by $|O_{C\cup T}|$ the number of computation output qubits in $C \cup T$. Similarly, $|O_C|$ is the number of output qubits in C. Without loss of generality, we impose that $|O_C|$ is minimal, in the sense that given the set of possible inputs and C, the space spanned by all possible outputs is the whole Hilbert space of dimension $2^{|O_C|}$. This is always possible as one can add a compression phase at the end of any non-minimal pattern.

Theorem 8 (Error-Correction Prevents Resource Blow-up). *Let C be a minimal MBQC pattern implementing a CPTP map C. Let $C \cup T$ denote a trappified pattern implementing C obtained from $\boldsymbol{P}(\lambda)$. Further assume that Protocol 3 using $\boldsymbol{P}(\lambda)$ has negligible security error $\max(\epsilon, \nu)$ with respect to λ.*

If $|O_{C\cup T}|/|O_C| = 1$ for a non-negligible fraction of trappified canvases $T \in \boldsymbol{P}(\lambda)$, then the overhead $|G_{\boldsymbol{P}(\lambda)}|/|G_C|$ is super-polynomial in λ.

The usefulness of this theorem comes from the contra-positive statement. Achieving exponential verifiability with a polynomial overhead imposes that $|O_{C\cup T}|/|O_C| > 1$ for an overwhelming fraction of the trappified patterns. This means that the computation is at least partially encoded into a larger physical Hilbert space, which then serves to actively perform some form of error-correction.

5 Correctness and Security Amplification for Classical Input-Output Computations

We now construct a generic compiler to boost the properties of trappified schemes in the case of classical inputs. This compiler is a direct application of the results from the previous section regarding the requirement of error-correction since it uses a classical repetition code to protect the computation from low-weight bit-flips. It works by decreasing the set of errors which are detected and increasing the set of errors to which the trappified scheme is insensitive. These errors then can be corrected via a recombination procedure, which in the classical case can be as simple as a majority vote. The proofs of the results from this section can be found in Appendix D.

Classical Input-Output Trappified Scheme Compiler. Theorem 6 presents a clear objective for traps: they should (i) detect harmful deviations while being insensitive to harmless ones. Yet, a trap in a trappified pattern cannot detect deviations happening on the computation part of the pattern itself. To achieve exponential verifiability, one further needs to ensure that there are sufficiently many trappified patterns so that it is unlikely that a potentially harmful deviation hits only the computation part of the pattern, and that it is detected with high probability when it hits the rest. This is best stated by Theorem 8, which imposes to (ii) error-protect the computation so that hard-to-detect deviations are harmless while remaining harmful errors are easy to detect. Additionally, one further needs to (iii) find a systematic way to insert traps alongside computation patterns to generate these exponentially many trappified patterns. Ideally, we would like to be able to design and analyse points (i), (ii) and (iii) independently from one another as much as possible. We show here a general way of performing this decomposition given slight constraints on the client's desired computation.

It is based on the realisation that if the client has d copies of its inputs – which is always possible whenever the inputs are classical – it can run d times its desired computation by repeating d times the desired pattern C on graph G sequentially or in parallel. If the output is classical, it is then naturally protected by a repetition code of length d and the result of the computation can be obtained through a majority vote. These d executions are called *computation rounds*. To detect deviations, the client needs to run s additional rounds which contain only traps. More precisely, each of these *test rounds* is a pattern run on the same graph G so that it is blind-compatible with C (see Definition 7). The collections of these s test rounds themselves constitute trappified canvases according to Definition 3, where acceptance is conditioned to less than w test rounds failures. Now, because computation rounds and test rounds are executed using blind-compatible patterns on the graph G, the trap insertion (iii) can be achieved by interleaving at random the s test rounds with the d computation rounds.

These steps, which are a generalisation of the technique from [20], are formalised in the following definition.

Definition 12 (Amplified Trappification Compiler). *Let $\boldsymbol{P}$ be trappified scheme on a graph $G = (V, E)$, and let $d, s \in \mathbb{N}$, $n = d + s$ and $w \in [s]$. Let $\mathfrak{C}$ be the class of computations with classical inputs that can be evaluated by an MBQC pattern on G using an order $\preceq_G$ which is compatible with the order $\preceq_{\boldsymbol{P}}$ induced by $\boldsymbol{P}$. We define the Amplified Trappification Compiler that turns $\boldsymbol{P}$ into a trappified scheme $\boldsymbol{P}'$ on G^n for computation class $\mathfrak{C}$ as follows:*

- *The trappified canvases $T' \in \boldsymbol{P}'$ and their distribution is given by the following sampling procedure:*
  1. *Randomly choose a set $S \subset [n]$ of size s. These will be the test rounds;*
  2. *For each $j \in S$, independently sample a trappified canvas T_j from the distribution of $\boldsymbol{P}$.*
- *For each trappified canvas T' defined above and an output $t = (t_j)_{j \in S}$, the output of the decision function τ' is obtained by thresholding over the outputs*

of the decision functions τ_j of individual trappified canvases. More precisely:

$$\tau'(t) = 0 \text{ if } \sum_{j \in S} \tau_j(t_j) < w, \text{ and } 1 \text{ otherwise} \tag{5}$$

- *The partial ordering of vertices of G^n in $\boldsymbol{P}'$ is given by the ordering $\preceq_G$ on each copy of G.*
- *Let $\mathsf{C} \in \mathfrak{C}$ and C the pattern on G which implements the computation C. Given a trappified canvas $T' \in \boldsymbol{P}'$, the embedding algorithm $E_{\mathfrak{C}}$ sets to C the pattern of the d graphs that are not in S.*

Boosting Detection and Insensitivity. The next theorem relates the parameters d, s, w with the deviation detection capability of the test rounds, thus showing that not only (i), (ii) and (iii) can be designed separately, but also analysed separately with regards to the security achieved by the protocol.

Theorem 9 (From Constant to Exponential Detection and Insensitivity Rates). *Let $\boldsymbol{P}$ be a trappified scheme on graph G which ϵ-detects the error set $\mathcal{E}_1$, is δ-insensitive to $\mathcal{E}_2$ and perfectly insensitive to $\{\mathsf{I}\}$. For $d, s \in \mathbb{N}$, $n = d + s$ and $w \in [s]$, let $\boldsymbol{P}'$ be the trappified scheme resulting from the compilation defined in Definition 12.*

For $\mathsf{E} \in \mathcal{G}_{V^n}$, let $\mathrm{wt}(\mathsf{E})$ be defined as the number of copies of G on which E does not act as the identity. We define $\mathcal{E}_{\geq k, \mathcal{F}} = \{\mathsf{E} \in (\mathcal{F} \cup \{\mathsf{I}\})^n \mid \mathrm{wt}(\mathsf{E}) \geq k\}$, and $\mathcal{E}_{\leq k, \mathcal{F}}$ analogously.

Let $k_1 > nw/(s\epsilon)$ and $k_2 < nw/(s\delta)$. Then, $\boldsymbol{P}'$ ϵ'-detects $\mathcal{E}_{\geq k_1, \mathcal{E}_1}$ and is δ'-insensitive to $\mathcal{E}_{\leq k_2, \mathcal{E}_2}$ where:

$$\epsilon' = \min_{\chi \in \left[0, \frac{k_1}{n} - \frac{w}{s\epsilon}\right]} \exp\left(-2\chi^2 s\right) + \exp\left(-2\frac{\left(\left(\frac{k_1}{n} - \chi\right) s\epsilon - w\right)^2}{\left(\frac{k_1}{n} - \chi\right) s}\right), \tag{6}$$

$$\delta' = \min_{\chi \in \left[0, \frac{w}{s\delta} - \frac{k_2}{n}\right]} \exp\left(-2\chi^2 s\right) + \exp\left(-2\frac{\left(\left(\frac{k_2}{n} + \chi\right) s\delta - w\right)^2}{\left(\frac{k_2}{n} + \chi\right) s}\right). \tag{7}$$

The consequence of the above theorem is that whenever the trappified schemes are constructed by interleaving computation rounds with test rounds chosen at random from a given set, the performance of the resulting protocol implementing SDQC crucially depends on the ability of these test rounds to detect harmful errors. Therefore, when using the compiler, optimisation of the performance is achieved by focussing only on designing more efficient test rounds. This is addressed in Section 6.

Remark 5. Note that we do not make use in Definition 12 of the embedding function or computation class associated with the trappified scheme $\boldsymbol{P}$. In fact the initial scheme can even consist of pure traps as described in Remark 3. This is the case for the schemes described in the next sections. If each trappified scheme used for tests can also embed the client's computation of interest, it is possible to use the alternative parallel repetition compiler presented in Appendix F which has no separate computation rounds.

Correctness Amplification via Majority Vote. Theorem 9 has given detection and insensitivity errors that are negligible n. In order to recover exponential verifiability, we must now also make the correctness error negligible in n. To this end, we recombine the multiple computation rounds into a single final result so that error of weight lower than k_2 are corrected.

Here, $\mathfrak{C}$ is the class of BQP computations that can be implemented on G, which implies that the failure probability for obtaining the correct result is c, below and bounded away from $1/2$. Then, we define $\boldsymbol{V}$ from the compiled $\boldsymbol{P}'$ by requiring that the input subspace is symmetric with respect to exchanging computation rounds – i.e. all computation rounds have the same inputs – and by defining the output subspace as the bitwise majority vote of computation round outputs. Intuitively, if it is guaranteed that the fraction of all rounds affected by a possibly harmful deviation is less than $(2c-1)/(2c-2)$ then the output of $\boldsymbol{V}$ will yield the correct result of the computation. This is because, in the large n limit, out of the d computation rounds a fraction c will be incorrect due to the probabilistic nature of the computation itself. Consequently, to maintain that more than $1/2$ the computation rounds yield the correct result so that the majority vote is able to eliminate the suprious results, the fraction f of computation rounds that the deviation can affect must satisfy $(1-c)(1-f) > c + (1-c)f$, that is $f < (2c-1)/(2c-2)$. Due to the blindness of the scheme, it is enough to impose that no more than a fraction $(2c-1)/(2c-1)$ of the n rounds is affected by the deviation to obtain the desired guarantee on the computation rounds with high probability.

Theorem 10 (Exponential Correctness from Majority Vote). *Let $\boldsymbol{T}$ be a trappified scheme on graph G which is perfectly correct on $\{\mathsf{I}\}$, for computations $\mathfrak{C} = \mathsf{BQP} \cap \mathfrak{G}$ where $\mathfrak{G}$ is the set of MBQC computations which can be performed on graph G. For $d, s \in \mathbb{N}$ and $n = d + s$, let $\boldsymbol{V}$ be the trappified scheme obtained through the compiler of Definition 12 and let the input subspace $\Pi_{I,C}$ be symmetric with respect to exchanging computation rounds. The output subspace $\Pi_{O,C}$ is defined as the concatenation of the (classical) outputs of all computation rounds and the decoding algorithm $\mathsf{D}_{O,C}$ is the bitwise majority vote of computation rounds outputs from the d computations.*

Let c be the bounded error of BQP computations and $k < \frac{2c-1}{2c-2}n$. Then, $\boldsymbol{V}$ is ν-correct on $\mathcal{E}_{\leq k, \mathcal{G}_V}$ for

$$\nu \leq \exp\left(-2\left(1 - \frac{2c-1}{2c-2} + \varphi - \epsilon_1\right) d\epsilon_2^2\right), \tag{8}$$

with

$$\frac{1}{2} - \left(\frac{2c-1}{2c-2} - \varphi + \epsilon_1\right) = (c + \epsilon_2)\left(1 - \frac{2c-1}{2c-2} + \varphi - \epsilon_1\right) \tag{9}$$

and $\varphi, \epsilon_1, \epsilon_2 > 0$. Thus ν is exponentially small in n if d/n is constant.

To conclude this section, we obtain simultaneous negligibility for detection, insensitivity and correctness errors by combining the conditions from Theorems

9 and 10:

$$\begin{aligned} & w = \left(\frac{2c-1}{2c-2} - \varphi - \chi\right) s(1-p) \\ & 0 < \varphi < \frac{2c-1}{2c-2}, \quad 0 < \chi < \frac{2c-1}{2c-2} - \varphi, \quad 0 < \epsilon_1 < \varphi \\ & \frac{1}{2} - \frac{2c-1}{2c-2} - \epsilon_1 = (c+\epsilon_2)\left(1 - \frac{2c-1}{2c-2} - \epsilon_1\right). \end{aligned} \tag{10}$$

Under these conditions, Theorem 6 yields an exponentially secure verification protocol using the trappified scheme $\boldsymbol{V}$.

Finally, while a simple majority vote is sufficient to recombine the computations in the classical case, finding such a distillation procedure in the quantum case is left as an open question.

6 New Optimised Trappified Schemes from Stabiliser Testing

In this section we demonstrate how the various tools and techniques introduced earlier can be combined to design trappified schemes that provide efficient and robust verifiability. To achieve this, we use Remark 2 and Lemma 2 to construct a trappified scheme $\boldsymbol{T}$ based on stabiliser testing with a constant detection error. Here we again focus on classical-input classical-output computations. Theorems 9 and 10 show that it is sufficient in this case to focus on designing test rounds, with the compiler from Definition 12 and majority vote then boosting the detection, insensitivity and correctness.

In the process, we show a close correspondence between prepare-and-send protocols derived from [10], and protocols based on stabiliser tests following [24]. This broadens noticeably the possibilities for designing new types of trappified patterns beyond those which are used by existing prepare-and-send protocols. It also allows to transfer existing protocols based on stabiliser testing from the non-communicating multi-server setting to the prepare-and-send model, thus lowering the assumptions of these protocols and making them more readily implementable and practical. We show in later subsections how to use the compiler results together with these new possibilities to optimise the current state-of-the-art protocol of [20].

6.1 Trappified Schemes from Subset Stabiliser Testing

Given $G = (V, E)$ and a partial order $\preceq_G$ on V, the first step for constructing a verification protocol for computations on G is to detect deviations from the server. To this end, we recall that any action from the server can be always be viewed as first performing the unitary part of Protocol 2 followed by a pure deviation that is independent from the computation delegated to the server (see Section 4). To be constructive and build traps that can be easily computed and checked by the client, we impose in this section that the outcomes of trappified canvases are deterministic and that they accept with probability 1 for honest executions of the protocol.

We first focus on the simplest case of deterministic functions, where the decision algorithm τ for the trappified canvas is such that $\tau(t) = t_i$ where t_i is measurement outcome of qubit i. In other words the test round accepts if the outcome $t_i = 0$, which corresponds to obtaining outcome $|0\rangle$ for qubit i, while all other measurements outcomes t_j for $j \neq i$ are ignored.[9]

For the outcome of the trappified canvas to be deterministic, qubit i must be equal to $|0\rangle$ in absence of deviations before the computational basis measurement. In other words, the state of i is an eigenstate of Z_i. By commuting Z_i towards the initialisation of the qubits – through the Hadamard gate and the entangling operations defined by the graph G, we conclude that determinism and acceptance of deviation-less test rounds implies that the initial state of the qubits before running the protocol is an eigenstate of $\mathsf{X}_i \bigotimes_{j\in N_G(i)} \mathsf{Z}_j = \mathsf{S}_i$.

The following lemma explains how to prepare a single-qubit tensor product state stabilised by such given Pauli operator.

Lemma 3 (Tensor Product Preparation of a State in a Stabiliser Subspace). *Let* P *be an element of the Pauli group over* N *qubits, such that* $\mathsf{P}^2 \neq -\mathsf{I}$. *Then, there exists* $|\psi\rangle = \bigotimes_{i=1}^N |\psi_i\rangle$ *such that* $|\psi\rangle = \mathsf{P}|\psi\rangle$, *and* $\forall i, |\psi_i\rangle \in \{|0\rangle, |+\rangle, |+_{\pi/2}\rangle\}$.

The proofs of all Lemmas in this Section are presented in Appendix E One can further note that the above lemma also holds for a set $\mathcal{R}$ of Pauli operators if, for all $\mathsf{P}, \mathsf{Q} \in \mathcal{R}$ and $i \in V$, either $\mathsf{P}(i) = \mathsf{Q}(i)$ or one acts as the identity on i. We call this the *no-overlap condition*.

Now take $\mathcal{R}$ a set of Pauli operators generating the stabiliser group of $|G\rangle$, and $\{\mathcal{R}^{(k)}\}_j$ a collection of subsets of $\mathcal{R}$ that such that each $\mathcal{R}^{(k)}$ satisfies the no-overlap condition and $\cup_k \mathcal{R}^{(k)} = \mathcal{R}$ – note that $\mathcal{R}$ need not be a minimal set of generators. We then construct a set of trappified canvases $T^{(k)}$ which have V as their input set and for which all qubits are measured in the X basis. They only differ in the prepared input states, each being prescribed by Lemma 3 for the stabilisers in $\mathcal{R}^{(k)}$ – that is qubits are prepared in an X, Y or Z eigenstate each time one of the Pauli operator in $\mathcal{R}^{(k)}$ is respectively X, Y or Z for this qubit, and chosen arbitrarily to be X eigenstates elsewhere. As above, the computation defined by the pattern where all qubits are measured in the X basis amounts to measuring the stabiliser generators S_i. The output distribution $\mathcal{T}^{(k)}$ can be computed given the prepared input state for $T^{(k)}$ using elementary properties of stabiliser states. But for our purposes, it is sufficient to construct the decision function $\tau^{(k)}$. This can be done by noting that for all $\mathsf{P} \in \mathcal{R}^{(k)}$, there is a unique binary vector $\{p_i\}_i$ such that $\mathsf{P} = \prod_i \mathcal{S}_i^{p_i}$. This, in turn, implies that $\mathcal{T}^{(k)}$ is such that $\bigoplus_i p_i t_i = 0$ where t_i is the outcome of the measurement of the i-th qubit in

[9] Recall that throughout the paper, our convention is to view rotated $\{|\pm_\theta\rangle\}$ measurements as Z rotations followed by a Hadamard gate and a measurement in the computational basis.

the X basis. Therefore, we define

$$\tau^{(k)}(t) = \bigwedge_{\mathsf{P} \in \mathcal{R}^{(k)}} \left(\bigoplus_i p_i t_i = 0 \right), \tag{11}$$

which reconstructs the measurement outcomes of stabilisers in $\mathcal{R}^{(k)}$ from the measurements outcomes of operators S_i. The function $\tau^{(k)}(t)$ will accept whenever the measurement outcomes of all stabilisers in $\mathcal{R}^{(k)}$ are zero. We denote by $\mathcal{E}_1^{(k)}$ the set of Pauli deviations that are perfectly detected by $T^{(k)}$ and $\mathcal{E}_2^{(k)} = \mathcal{G}_V \setminus \mathcal{E}_1^{(k)}$ the set of deviations to which $T^{(k)}$ is perfectly insensitive.

Now, using Remark 2 and Lemma 2, the trappified canvases $T^{(k)}$ can be composed with equal probability p to obtain a trappified scheme $\boldsymbol{T}$. We then consider the sets of all Pauli deviations $\mathcal{E}_1 = \bigcup_k \mathcal{E}_1^{(k)}$ and $\mathcal{E}_2 = \bigcup_k \mathcal{E}_2^{(k)} = \mathcal{G}_V$. We conclude that the scheme $\boldsymbol{T}$ then $(1-p)$-detects $\mathcal{E}_1$ and is $(1-p)$-insensitive to $\mathcal{G}_V$. Note that these values are upper-bounds, with equality being achieved if there is no overlap in the set of errors which each canvas can detect.

The scheme $\boldsymbol{T}$ therefore detects all possibly harmful deviations with finite probability, and is partly insensitive to all deviations – i.e. both harmless and harmful – that can affect computations in $\mathfrak{C}$.

Two Paths Towards Scheme Optimisations. At first glance, the main goal to optimise such schemes seems to be to lower as much as possible the number of subsets of stabilisers $\mathcal{R}^{(k)}$ which cannot be tested at the same time. Each such subset of stabilisers needs a different canvas $T^{(k)}$ to test for it, and the probability p increases with a lower number of canvases. An increase in p automatically decreases the detection and insensitivity errors. These in turn appear in the exponential bounds from Theorem 9, meaning that even a slight decrease greatly influences the total security for a given number of repetitions, or equivalently the number of repetitions required to achieve a given security level.

However this is the case only if each test detects a set of errors disjoint from those detected by the other sets. Another way to increase the probability of detection is to increase the coverage of each canvas by increasing the number of stabiliser errors which each can detect. In this case, the sets can be made to overlap and the detection probability can be lowered below the upper-bound of $1-p$. We explore both approaches in the next two subsections and give in Appendix G a general process for systematising this optimisation with different constraints.

6.2 Standard Traps

The simplest application of Lemma 3 is to prepare qubit i_0 as an eigenstate of X, while its neighbours in the graph are prepared as an eigenstate of Z. This setup can detect all deviations which do not commute with the Z_{i_0} measurements of i_0. Here, the reader familiar with the line of work following [10] note that we have recovered their single-qubit traps: single qubits prepared in the $X-Y$ plane and surrounded by dummy $|0\rangle$ or $|1\rangle$ qubits.

Additionally, within each test round, it is possible to include several such atomic traps as long as their initial states can be prepared simultaneously – i.e. they can at most overlap on qubits that need to be prepared as eigenstates of Z. More precisely, take H to be an independent set of vertices from G (see Definition 15). We define the set of stabilisers associated to H as $\mathcal{R}_H = \{\mathsf{S}_i\}_{i\in H}$. Such sets naturally follow the no-overlap condition since H is an independent set and therefore if $i \neq j$, $\mathsf{S}_i(j) = \mathsf{S}_j(i) = \mathsf{I}$ and both stabilisers are equal to either Z or I for all qubits different from i or j. This is the extreme case where all stabilisers in $\mathcal{R}_H$ have a single component when decomposed in the generator set $\{\mathsf{S}_i\}$. Following the same line of argument as above, in absence of deviation, the state of qubit i must be $|0\rangle$ for all $i \in H$ before the measurement, or equivalently, is an eigenstate of Z_i. Commuting these operators towards the initialisation of the qubits shows that the qubits in H must be prepared in the state $|+\rangle$, and $|0\rangle$ for qubits in $N_G(H)$. These qubits form the input set I_T of the trappified canvas T_H associated to the independent set H. Other qubits can be prepared in any allowed state. Its output locations O_T are the independent set H.

Using the formula from Equation 11 for set $\mathcal{R}_H$, we get $\tau(t) = \bigwedge_{i\in H} t_i$ for the decision algorithm. That is, the trappified canvas accepts whenever all outcomes Z measurements for qubits $i \in H$ are 0.

A trappified canvas T_H generated in this way depends only on the choice of independent set H. Such trappified canvases will be called *standard trap* in the remaining of this work.

Let $\{H^{(j)}\}_j$ be a set of independent sets. Since $\mathcal{R}_{H^{(j)}}$ contains all stabilisers S_i for $i \in H^{(j)}$, the sets $\mathcal{R}_{H^{(j)}}$ cover the generating set of stabiliser $\{\mathsf{S}_i\}_{i\in V}$ entirely if and only if each qubit $i \in V$ is in at least one of the independent sets $H^{(j)}$. Then one can conclude that all X and Y deviations have a non-zero probability of being detected, while I and Z deviations are never detected, but are harmless for classical output computations.

Optimising Standard Traps. The background in graph theory and graph colourings necessary for this section can be found in Appendix A.4.

The crucial parameter to optimise is the detection probability of individual test rounds with respect to X deviations. In other words, the performance of the scheme will vary depending on the choice of probability distribution over the independent set $\mathcal{I}(G)$ and the detection capability of each individual test round.

A test round, and therefore its corresponding trappified canvas, will detect a Pauli error if and only if at least one of the $|+\rangle$-states is hit by a local X or Y deviation.

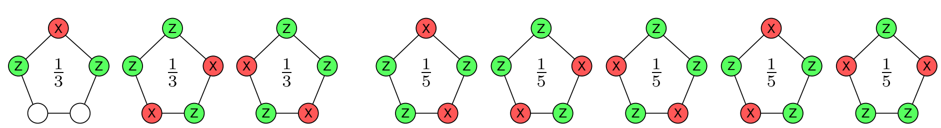
Lemma 4 (Detection Rate). *Let $G = (V, E)$ be an undirected graph. Let $\mathcal{D}$ be a probability distribution over $\mathcal{I}(G)$, giving rise to the trappified scheme $\boldsymbol{P}$ where every element of $\mathcal{I}(G)$ describes one trappified canvas. We define the* detection rate *of $\mathcal{D}$ over G as $p_{det}(\mathcal{D}) = 1 - \epsilon(\mathcal{D}) = \min_{\substack{M\subseteq V\\ M\neq\emptyset}} \Pr_{H\sim\mathcal{D}}[M \cap H \neq \emptyset]$. Then $\boldsymbol{P}$ $\epsilon(\mathcal{D})$-detects the error set $\mathcal{E} = \{\mathsf{I},\mathsf{X},\mathsf{Y},\mathsf{Z}\}^{\otimes V} \setminus \{\mathsf{I},\mathsf{Z}\}^{\otimes V}$.*

In the definition above, H corresponds to a choice of test round, while M is the set of qubits that are affected by to-be-detected X and Y deviations. To obtain the lowest overhead, the distribution $\mathcal{D}$ should be chosen such that it maximises the detection probability $1-\epsilon(\mathcal{D})$ for a given graph G. It can be shown that the best achievable detection rate by standard traps for a graph G lie in the interval $\left[\frac{1}{\chi(G)}, \frac{1}{\omega(G)}\right]$, where $\chi(G)$ and $\omega(G)$ are respectively the chromatic number and the clique number of G. The protocol of [20] in particular is designed with security bounds depending on the chromatic number of the underlying graph. Note that the two graph invariants $\chi(G)$ and $\omega(G)$ are dual in the sense that they are integer solutions to dual linear programs and the gap between these two values can be large (see Lemma 8). It turns out that both bounds can be improved to depend on the solutions of the relaxations of the respective linear programs. This closes the integrality gap between the chromatic number and the clique number.

Lemma 5. *For every (non-null) graph G there exists a distribution $\mathcal{D}$ over $\mathcal{I}(G)$ such that $p_{det}(\mathcal{D}) \geq \frac{1}{\chi_f(G)}$, with $\chi_f(G)$ the fractional chromatic number of G. Further, for every distribution $\mathcal{D}'$ over $\mathcal{I}(G)$ it holds that $p_{det}(\mathcal{D}') \leq \frac{1}{\omega_f(G)}$, with $\omega_f(G)$ the fractional clique number of G.*

As a consequence, this shows that the protocol described in [20], which is the current state-of-the-art, can sometimes be improved by constructing additional test rounds that would allow to have a probability of detection greater than the reported $1/\chi(G)$. In fact, this proves that the best possible detection rate by standard traps is equal to $1/\chi_f(G)$ since $\chi_f(G) = \omega_f(G)$ by Lemma 11. This is achieved precisely by choosing the set of possible tests to be a fractional colouring of the graph.

Example 1. Let $G = (V, E)$ be the cycle graph on 5 nodes with $V = \{0, 1, 2, 3, 4\}$. An optimal proper 3-colouring of G is given by $(\{0, 2\}, \{1, 3\}, \{4\})$, which gives rise to a standard trap with detection rate $1/3$. However, this may be further improved using Lemma 5 and the fact that $\chi_f(G) = 5/2$. A standard trap with the optimal detection rate of $2/5$ is given by the uniform distribution over the set $\{\{0, 2\}, \{1, 3\}, \{2, 4\}, \{0, 3\}, \{1, 4\}\}$.



(a) Trap distribution based on an optimal colouring of G.

(b) Trap distribution based on an optimal fractional colouring of G.

Fig. 1: Traps on the cycle graph G with 5 nodes from Example 1.

Yet, this leaves a dependency of the protocol's efficiency on graph invariants, meaning that depending on the chosen computation, the protocol could perform poorly. The next section shows how to overcome this obstacle, as long as the client is willing to use more generalised traps.

6.3 General Traps

Above, the trappified canvases we obtained are a consequence of determinism, insensitivity to harmless deviations and a restriction on the subsets H, constrained to be independent. To construct *general traps*, we simply remove this last requirement and define instead $\mathcal{R}_H = \{\prod_{i \in H} \mathsf{S}_i\}$. Using Equation 11, τ is then the parity of measurement outcomes for qubits from H, i.e. $\tau(t) = \bigoplus_{i \in H} t_i$. This means that to accept the execution of such trappified canvas, the state of the qubits $i \in H$ needs to be in the $+1$ eigenspace of the operator $\bigotimes_{i \in H} \mathsf{Z}_i$. This is the other extreme case since there is only a single stabiliser in the set $\mathcal{R}_H$.

Commuting this operator to the initialisation imposes to prepare a $+1$ eigenstate of $\bigotimes_{i \in H_{even}} \mathsf{X}_i \bigotimes_{j \in H_{odd}} \mathsf{Y}_j \bigotimes_{k \in N_G^{odd}(H)} \mathsf{Z}_k$, where H_{even} (resp. H_{odd}) are the qubits of even (resp. odd) degree within H, and $k \in N_G^{odd}(H)$ means k is in the odd neighbourhood of H. Again, applying Lemma 3 allows us to find in the eigenspace of this operator a state that can be obtained as a tensor product of single-qubit states, simply by looking at the individual Paulis from the operator $\prod_{i \in H} \mathsf{S}_i$. It is easy to see that this trappified canvas detects all deviations that anti-commute with $\bigotimes_{i \in H} \mathsf{Z}_i$, that is deviations that have an odd number of X or Y for qubits in H. Varying the sets H allows to construct a trappified scheme which detects all possible deviations containing any number of X or Y with a constant probability.

Optimising General Traps. General traps are based on test rounds defined by a set $H \subseteq V$ of qubit locations. It accepts whenever the parity of outcomes of Z-measurements on the qubits of H is even. Here the testing set H can be chosen freely and does not need to be independent as in the construction of standard traps.

Lemma 6 (General Stabiliser-Based Trappified Scheme). *Let $\boldsymbol{P}$ be the trappified scheme defined by sampling uniformly at random a non-empty set $H \subseteq V$ and preparing the trappified canvas associated to $\mathcal{R}_H = \{\prod_{i \in H} \mathsf{S}_i\}$. Then $\boldsymbol{P}$ $1/2$-detects the error set $\mathcal{E} = \{\mathsf{I}, \mathsf{X}, \mathsf{Y}\}^{\otimes V} \setminus \{\mathsf{I}^{\otimes V}\}$.*

As a conclusion, we obtain that the probability of detection for this scheme is equal to $1/2$, which is independent of the graph G, and generally will beat the upper bound obtained in the previous section through standard traps.

7 Discussion and Future Work

We uncovered a profound correspondence between error-detection and verification that applies and unifies all previous trap-based blind verification schemes in the prepare-and-send MBQC model, which covers the majority of proposed protocols from the literature. In addition, all results mentioned here also apply to receive-and-measure MBQC protocols via the recent equivalence result from [28]. On the theoretical side, it provides a direct and generic composable security proof of these protocols in the AC framework, which also gives the first direct and explicit proof of composability if the original VBQC protocol [10]. We also formally showed that error-correction is required if one hopes to have negligible correctness and security errors with polynomial overhead when comparing unprotected and unverified computations to their secure counterparts. On a practical side, this correspondence can be used to increases the tools available to design, prove the composable security, and optimise the performance of new protocols. To exemplify these new possibilities, we described new protocols that improve the overhead of state-of-the-art verification protocols, thus making them more appealing for experimental realisation and possibly for integration into future quantum computing platforms.

The uncovered connection between error-detection and verification raises new questions such as the extent to which it is possible to infer from the failed traps what the server has been performing. Additionally, Theorem 8 implies that some form of error-correction is necessary to obtain exponential correctness. Yet, our protocol [20] shows that sometimes classical error-correction is enough, thereby raising the question of understanding what are the optimal error-correction schemes for given classes of computations that are to be verified. Finally, we strongly suspect the link between error detection and verification can be further developed and yield new trappified schemes with not only more efficient implementations but also additional capabilities.

References

1. Aaronson, S.: The Scott Aaronson 25.00$ Prize. `http://www.scottaaronson.com/blog/?p=284` (October 2007), accessed: Jan. 30 2015
2. Aharonov, D., Ben-Or, M., Eban, E.: Interactive proofs for quantum computations. In: Proceedings of Innovations in Computer Science 2010. pp. 453–. ICS2010 (2010)
3. Aharonov, D., Ben-Or, M., Eban, E., Mahadev, U.: Interactive proofs for quantum computations. arxiv:1704.04487 (2017), updated and corrected version of arxiv:0810.5375
4. Broadbent, A.: How to verify a quantum computation. arXiv preprint arXiv:1509.09180 (2015)
5. Broadbent, A., Fitzsimons, J., Kashefi, E.: Universal blind quantum computation. In: IEEE (ed.) 50th Annual IEEE Symposium on Foundations of Computer Science (2009)
6. Danos, V., Kashefi, E.: Determinism in the one-way model. Phys. Rev. A **74**, 052310 (Nov 2006). `https://doi.org/10.1103/PhysRevA.74.052310`, `http://link.aps.org/doi/10.1103/PhysRevA.74.052310`
7. Danos, V., Kashefi, E., Panangaden, P.: The measurement calculus. J. ACM **54**(2) (Apr 2007). `https://doi.org/10.1145/1219092.1219096`, `http://doi.acm.org/10.1145/1219092.1219096`
8. Dunjko, V., Fitzsimons, J.F., Portmann, C., Renner, R.: Composable security of delegated quantum computation. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014. pp. 406–425. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
9. Ferracin, S., Kapourniotis, T., Datta, A.: Reducing resources for verification of quantum computations. Physical Review A **98**(2), 022323 (2018)

10. Fitzsimons, J.F., Kashefi, E.: Unconditionally verifiable blind quantum computation. Phys. Rev. A **96**, 012303 (Jul 2017). https://doi.org/10.1103/PhysRevA.96.012303, https://link.aps.org/doi/10.1103/PhysRevA.96.012303
11. Fitzsimons, J.F., Hajdušek, M., Morimae, T.: Post hoc verification of quantum computation. Phys. Rev. Lett. **120**, 040501 (Jan 2018). https://doi.org/10.1103/PhysRevLett.120.040501, https://link.aps.org/doi/10.1103/PhysRevLett.120.040501
12. Gennaro, R.: Verifiable outsourced computation: A survey. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 313–313 (2017)
13. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
14. Hangleiter, D., Kliesch, M., Schwarz, M., Eisert, J.: Direct certification of a class of quantum simulations. Quantum Science and Technology **2**(1), 015004 (feb 2017). https://doi.org/10.1088/2058-9565/2/1/015004, https://doi.org/10.1088/2058-9565/2/1/015004
15. Hayashi, M., Morimae, T.: Verifiable measurement-only blind quantum computing with stabilizer testing. Phys. Rev. Lett. **115**, 220502 (Nov 2015). https://doi.org/10.1103/PhysRevLett.115.220502, https://link.aps.org/doi/10.1103/PhysRevLett.115.220502
16. Hein, M., Eisert, J., Briegel, H.J.: Multiparty entanglement in graph states. Phys. Rev. A **69**, 062311 (Jun 2004). https://doi.org/10.1103/PhysRevA.69.062311, https://link.aps.org/doi/10.1103/PhysRevA.69.062311
17. Kapourniotis, T.: Efficient verification of universal and intermediate quantum computing. Ph.D. thesis, School of Informatics, University of Edinburgh (2016)
18. Kapourniotis, T., Dunjko, V., Kashefi, E.: On optimising quantum communication in verifiable quantum computing (2015), presented at AQIS'15 conference
19. Kashefi, E., Wallden, P.: Optimised resource construction for verifiable quantum computation. Journal of Physics A: Mathematical and Theoretical; preprint arXiv:1510.07408 (2017), http://iopscience.iop.org/10.1088/1751-8121/aa5dac
20. Leichtle, D., Music, L., Kashefi, E., Ollivier, H.: Verifying bqp computations on noisy devices with minimal overhead. Phys. Rev. X Quantum **2**(040302) (2021)
21. Mahadev, U.: Classical verification of quantum computations. In: Thorup, M. (ed.) 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018. pp. 259–267. IEEE Computer Society (2018). https://doi.org/10.1109/FOCS.2018.00033, https://doi.org/10.1109/FOCS.2018.00033
22. Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) Theory of Security and Applications. pp. 33–56. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
23. Maurer, U., Renner, R.: Abstract cryptography. In: Innovations in Computer Science. pp. 1 – 21. Tsinghua University Press (jan 2011), https://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/14.html
24. McKague, M.: Interactive proofs for BQP via self-tested graph states. Theory of Computing **12**(3), 1–42 (2016). https://doi.org/10.4086/toc.2016.v012a003, https://theoryofcomputing.org/articles/v012a003
25. Raussendorf, R., Harrington, J., Goyal, K.: Topological fault-tolerance in cluster state quantum computation. New Journal of Physics (2007)
26. Raussendorf, R., Briegel, H.J.: A one-way quantum computer. Phys. Rev. Lett. **86**, 5188–5191 (May 2001). https://doi.org/10.1103/PhysRevLett.86.5188, http://link.aps.org/doi/10.1103/PhysRevLett.86.5188

27. Vazirani, U.: Workshop on the computational worldview and the sciences (2007), http://users.cms.caltech.edu/~schulman/Workshops/CS-Lens-2/report-comp-worldview.pdf
28. Wiesner, F., Eisert, J., Pappa, A.: Equivalence in delegated quantum computing (2022). https://doi.org/10.48550/ARXIV.2206.07469, https://arxiv.org/abs/2206.07469
29. Xu, Q., Tan, X., Huang, R.: Improved resource state for verifiable blind quantum computation. Entropy **22**(9) (2020). https://doi.org/10.3390/e22090996, https://www.mdpi.com/1099-4300/22/9/996

Auxiliary Supporting Material

A Additional Preliminary Material

A.1 Notation

Throughout this work we will use the following notation

- For a set V, $\wp(V)$ is the powerset of V, the set of all subsets of V.
- For a set $B \subseteq A$, we denote by B^c the complement of B in A, where A will often be the vertex set of a graph and B a subset of vertices, usually input or output locations.
- For $n \in \mathbb{N}$, the set of all integers from 0 to n included is denoted $[n]$.
- For a real function $\epsilon(\eta)$, we say that $\epsilon(\eta)$ is *negligible in* η if, for all polynomials $p(\eta)$ and η sufficiently large, we have $\epsilon(\eta) \leq \frac{1}{p(\eta)}$.
- For a real function $\mu(\eta)$, we say that $\mu(\eta)$ is *overwhelming in* η if there exists a negligible $\epsilon(\eta)$ such that $\mu(\eta) = 1 - \epsilon(\eta)$.

A.2 Delegated MBQC Protocol

Protocol 4 Delegated MBQC Protocol

Client's Inputs: A measurement pattern $(G, I, O, \{\phi(i)\}_{i \in O^c}, f)$ and a quantum register containing the input qubits $i \in I$.

Protocol:

1. The Client sends the graph's description (G, I, O) to the Server;
2. The Client sends its input qubits for positions I to the Server;
3. The Server prepares $|+\rangle$ states for qubits $i \in I^c$;
4. The Server applies a CZ gate between qubits i and j if (i, j) is an edge of G;
5. The Client sends the measurement angles $\{\phi(i)\}_{i \in O^c}$ along with the description of f to the Server;
6. The Server measures the qubits $i \in O^c$ in the order defined by f in the rotated basis $\left|\pm_{\phi'(i)}\right\rangle$ where

$$s_X(i) = \bigoplus_{j \in S_X(i)} b(j), \; s_Z(i) = \bigoplus_{j \in S_Z(i)} b(j), \tag{12}$$

$$\phi'_i = (-1)^{s_X(i)} \phi(i) + s_Z(i)\pi, \tag{13}$$

   where $b(j) \in \{0, 1\}$ is the measurement outcome for qubit j, with 0 being associated to $\left|+_{\phi'(j)}\right\rangle$, and $S_X(i)$ (resp. $S_Z(i)$) is the X (resp. Z) dependency set for qubit i defined by $S_X(i) = f^{-1}(i)$ (resp. $S_Z(i) = \{j : i \in N_G(f(j))\}$);
7. The Server performs the correction $\mathsf{Z}^{s_Z(i)}\mathsf{X}^{s_X(i)}$ for output qubits $i \in O$, which it sends back to the Client.

To analyse the security of our protocol later, we will require the following Pauli Twirling Lemma as a way to decompose the actions of an Adversary in the blind protocol above. A Pauli twirl occurs when a random Pauli operator is applied (such as an encryption and decryption). 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. This has the effect of removing all off-diagonal factors from the operation sandwiched between the two applications of the random Pauli, thus making it a convex combination of Pauli operators.

Lemma 7 (Pauli Twirling).
Let ρ be an n-qubit mixed state and $\mathsf{Q}, \mathsf{Q}' \in \mathcal{P}_n$ two n qubit Pauli operators. Then, if $\mathsf{Q} \neq \mathsf{Q}'$, we have:

$$\sum_{\mathsf{P} \in \mathcal{P}_n} \mathsf{P}^\dagger \mathsf{Q} \mathsf{P} \rho \mathsf{P}^\dagger \mathsf{Q}'^\dagger \mathsf{P} = 0 \tag{14}$$

A.3 Abstract Cryptography

Abstract Cryptography (AC) is a security framework for cryptographic protocols that was introduced in [23,22]. The focus of the AC framework is to provide general composability. In this way, protocols that are separately shown to be secure within the framework can be composed in sequence or in parallel while keeping a similar degree of security. See [8] for further details.

On an abstract level, the AC framework considers resources and protocols. While a resource provides a specified functionality, protocols are essentially instructions how to construct resources from other resources. In this way, this framework allows the expansion of the set of available resources while ensuring general compatibility.

Technically, a quantum protocol π with N honest parties is described by $\pi = (\pi_1, \ldots, \pi_N)$, where the combined actions of party i, denoted π_i, are called the converter of party i and consist in the quantum case of a sequence of efficiently implementable CPTP maps. A resource has interfaces with the parties that are allowed to exchange states with it. During its execution, it waits for all input interfaces to be initialised, then applies a CPTP map to all interfaces and its internal state, and finally transmits the states in the output interfaces back to the appropriate parties. This process may be repeated multiple times. Entirely classical resources can be enforced by immediate measurements of all input registers and the restriction of the output to computational basis states.

AC security is entirely based on the indistinguishability of resources. A protocol is considered to be secure if the resource which it constructs is indistinguishable from an ideal resource which encapsulates the desired security properties. Two resources with the same number of interfaces are called indistinguishable if no algorithm, called the distinguisher, can guess with good probability the resource that it is given access to. During this process, the distinguisher obtains access to all of the resource's interfaces at the same time.

Definition 13 (Indistinguishability of Resources). *Let $\epsilon > 0$ and $\mathcal{R}_1$ and $\mathcal{R}_2$ be two resources with same input and output interfaces. Then, these resources are called ϵ-statistically-indistinguishable, denoted $\mathcal{R}_1 \underset{stat,\epsilon}{\approx} \mathcal{R}_2$, if for all (unbounded) distinguishers $\mathcal{D}$ it holds that*

$$\left| \Pr[b = 1 \mid b \leftarrow \mathcal{D}\mathcal{R}_1] - \Pr[b = 1 \mid b \leftarrow \mathcal{D}\mathcal{R}_2] \right| \leq \epsilon \tag{15}$$

Analogously, $\mathcal{R}_1$ and $\mathcal{R}_2$ are said to be computationally indistinguishable if this holds for all quantum polynomial-time distinguishers.

With this definition in mind, the correctness of a protocol is captured by the indistinguishability of the resource constructed by the protocol from the ideal resource when all parties are honest, i.e. they use their respective converters as specified by the protocol. The security of the protocol against a set of malicious and collaborating parties is given by the indistinguishability of the constructed resource where the power of the distinguisher is extended to the transcripts of the corrupted parties. This is formally captured by Definition 14.

Definition 14 (Construction of Resources).
Let $\epsilon > 0$. 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 $\pi_{M^c}\mathcal{R} \underset{stat,\epsilon}{\approx} \mathcal{S}\sigma_M$*

Analogously, computational correctness and security is given for computationally bounded distinguishers as in Definition 13, and with a quantum polynomial-time simulator σ_M.

This finally allows us to formulate the General Composition Theorem at the core of the Abstract Cryptography framework.

Theorem 11 (General Composability of Resources [23, Theorem 1]).
Let $\mathcal{R}$, $\mathcal{S}$ and $\mathcal{T}$ be resources, α, β and id be protocols (where protocol id does not modify the resource it is applied to). Let $\circ$ and $\mid$ 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 $(\alpha \mid \mathsf{id})(\mathcal{R} \mid \mathcal{T}) \underset{stat,\epsilon_\alpha}{\approx} (\mathcal{S} \mid \mathcal{T})$.*

A combination of these two properties yields concurrent composability, where the distinguishing advantage accumulates additively as well. The following resource models the security of the UBQC Protocol 2. It leaks no information to the Server beyond a controlled leak, but allows the Server to modify the output by deviating from the Client's desired computation. The following theorem captures the security guarantees of the UBQC Protocol 2 in the Abstract Cryptography Framework, as expressed in [8].

Theorem 12 (Security of Universal Blind Quantum Computation).
The UBQC Protocol 2 perfectly constructs the Blind Delegated Quantum Computation Resource 2 for leak $l_{\rho_C} = (G, O, \preceq_G)$, where $\preceq_G$ is the ordering induced by the flow of the computation.

Resource 2 Blind Delegated Quantum Computation

Public Information: Nature of the leakage l_{ρ_C}.

Inputs:

- The Client inputs the classical description of a computation C from subspace $\Pi_{I,C}$ to subspace $\Pi_{O,C}$ and a quantum state ρ_C in $\Pi_{I,C}$.
- 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 $c = 1$, the Server has an additional input CPTP map F and state ρ_S.

Computation by the Resource:

1. If $e = 1$, the Resource sends the leakage l_{ρ_C} to the Server's interface.
2. If $c = 0$, it outputs $\mathsf{C}(\rho_C)$ at the Client's output interface. Otherwise, it waits for the additional input and outputs $\mathrm{Tr}_S(\mathsf{F}(\rho_{CS}))$ at the Client's interface.

A.4 Graph Colourings

In this section, we introduce graph colourings and recall some known related results that are useful to our theory.

Definition 15 (Independent Set). *Let $G = (V, E)$ be a graph. Then a set of vertices $t \subseteq V$ is called an* independent set *of G if*

$$\forall v_1, v_2 \in t : \{v_1, v_2\} \notin E. \tag{16}$$

The size of the largest independent set of G is called the independence number *of G and denoted by $\alpha(G)$. The set of all independent sets of G is denoted $I(G)$.*

Definition 16 (Graph Colouring). *Let $G = (V, E)$ be a graph. Then a collection of k pairwise disjoint independent sets $H_1, \ldots, H_k \subseteq V$ such that $\bigcup_{j=1}^{k} H_j = V$ is called a* (proper) k-colouring *of G. The smallest number $k \in \mathbb{N}_0$ such that G admits a k-colouring is called the chromatic number of G and denoted by $\chi(G)$.*

Definition 17 (Clique). *Let $G = (V, E)$ be a graph. Then a complete subgraph $C \subseteq V$ of size k is called a* k-clique *of G. The largest number $k \in \mathbb{N}_0$ such that G admits a k-clique is called the clique number of G and denoted by $\omega(G)$.*

Lemma 8. *For any graph G it holds that $\omega(G) \leq \chi(G)$. For any $n \in \mathbb{N}$, there exists a graph G_n such that $\chi(G_n) - \omega(G_n) \geq n$.*

Definition 18 (Fractional Graph Colouring). *Let $G = (V, E)$ be a graph. For $b \in \mathbb{N}$, a collection of independent sets $H_1, \ldots, H_k \subseteq V$, such that for all $v \in V : |\{1 \leq j \leq k \mid v \in H_j\}| = b$, is called a* k:b-colouring *of G. The smallest number $k \in \mathbb{N}_0$ such that G admits a* k:b-colouring *is called the b-fold chromatic number of G and denoted by $\chi_b(G)$. Since $\chi_b(G)$ is subadditive we can define the* fractional chromatic number *of G as*

$$\chi_f(G) = \lim_{b \to \infty} \frac{\chi_b(G)}{b} = \inf_{b \in \mathbb{N}} \frac{\chi_b(G)}{b}. \tag{17}$$

Note that k:1-colourings are k-colourings and therefore $\chi_1(G) = \chi(G)$ which in turn implies that for all $b \in \mathbb{N}$ it holds that

$$\chi_f(G) \leq \chi_b(G) \leq \chi(G). \tag{18}$$

Lemma 9. *Let $G = (V, E)$ be a graph. Then $\chi_f(G)$ equals the smallest number $k \in \mathbb{R}_0^+$ such that there exists a probability distribution $\mathcal{D}$ over the independent sets $\mathcal{I}(G)$ such that for all $v \in V$ it holds that*

$$\Pr_{H \leftarrow \mathcal{D}}[v \in t] \geq \frac{1}{k}. \tag{19}$$

Definition 19 (Fractional Clique). *Let $G = (V, E)$ be a graph. For $b \in \mathbb{N}$, a function $f : V \to \mathbb{N}_0$, such that for all $H \in \mathcal{I}(G) : \sum_{v \in H} f(v) \leq b$ and $\sum_{v \in V} = k$, is called a k:b-*clique *of G. The biggest number $k \in \mathbb{N}_0$ such that G admits a k:b-*clique *is called the b-fold clique number of G and denoted by $\omega_b(G)$. Since $\chi_b(G)$ is superadditive we can define the* fractional clique number *of G as*

$$\omega_f(G) = \lim_{b \to \infty} \frac{\omega_b(G)}{b} = \sup_{b \in \mathbb{N}} \frac{\omega_b(G)}{b}. \tag{20}$$

Note that k:1-cliques are k-cliques and therefore $\omega_1(G) = \omega(G)$ which in turn implies that for all $b \in \mathbb{N}$ it holds that

$$\omega(G) \leq \omega_b(G) \leq \omega_f(G). \tag{21}$$

Lemma 10. *Let $G = (V, E)$ be a graph. Then $\omega_f(G)$ equals the biggest number $k \in \mathbb{R}_0^+$ such that there exists a probability distribution $\mathcal{D}$ over the vertices V such that for all $H \in \mathcal{I}(G)$ it holds that*

$$\Pr_{v \leftarrow \mathcal{D}}[v \in H] \leq \frac{1}{k}. \tag{22}$$

Both the fractional clique number $\omega_f(G)$ and the fractional chromatic number $\chi_f(G)$ are rational-valued solutions to dual linear programs. By the strong duality theorem, the two numbers must be equal.

Lemma 11. *For any graph G it holds that $\omega_f(G) = \chi_f(G)$.*

B Examples for Constructs of Section 3

B.1 Partial Pattern for Computing

Let G be the $n \times m$ 2D-cluster graph – i.e. n-qubit high and m-qubit wide – and the ordering of the qubits starting in the upper-left corner, going down first then right. Such graph state is universal for MBQC [26]. There are many possible partial patterns that can be defined on such graph. For instance, consider a pattern Q that runs on a smaller $n' \times m'$ 2D-cluster graph. Then, one can define a partial pattern P on G as the top-left $(n'+1) \times (m'+1)$ subgraph. The set I_P is defined as the set I of Q together with all the qubits on the bottom row and right column. The input space corresponds to the Hilbert space of the input qubits of Q tensored with $|0\rangle$ for the qubits of the bottom row and right column. The output set O_P is the same set as in Q and $\Pi_{O,P}$ is the full Hilbert space of the output qubits. The measurement angles are the same as in Q for the corresponding qubits and set to be random for the bottom row and right column. The flow is the same as in Q, provided that the added $|0\rangle$ qubits have no dependent qubits. Because the added qubits are forced to be in the $|0\rangle$ state, this isolates a $n' \times m'$ 2D-cluster graph that can then be used to perform the same operations as in Q, thereby allowing to compute the same unitary, albeit using a larger graph, see Figure 2. Note that one can change the location of the $n' \times m'$ 2D-cluster graph used for the computation, as long as it is properly surrounded by qubits in the $|0\rangle$ state. This is done by defining the input subspace of the partial pattern to take that constraint into account.

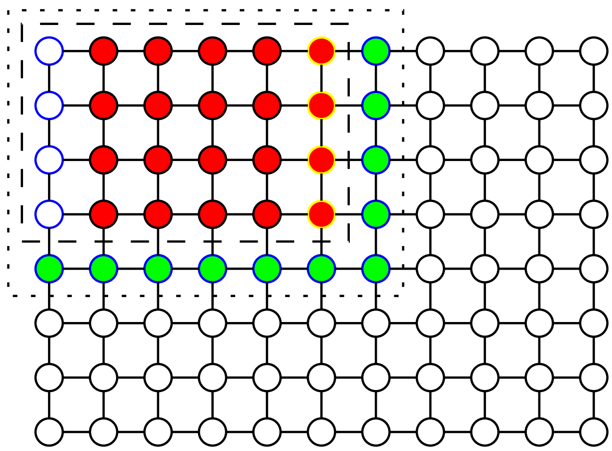


Fig. 2: Partial pattern for computing. The partial pattern is in the dashed box. Input qubits are surrounded in blue, output qubits in yellow. Red filled qubits are prepared in $|+\rangle$ while the green ones are prepared in $|0\rangle$. The green qubits define a subspace of the Hilbert space of the input qubits that guarantees that a 4×6 cluster state computation can be run inside the long-dashed box.

B.2 Canvas with a Single Standard Trap

Consider the $n \times m$ 2D-cluster graph and consider the partial pattern of Example B.1 where the subgraph is a 3×3 square – i.e. a single computation qubit surrounded by 8 $|0\rangle$ states. The input state is fixed to be $\sigma = |+\rangle \otimes |0\rangle^{\otimes 8}$ where $|+\rangle$ is the state of the central qubit, the others being the aforementioned peripheral ones. Because the central qubit is measured along the X-axis $\mathcal{T}$ is deterministic – the measurement outcome 0 corresponding to the projector $|+\rangle\langle+|$ has probability 1. The accept function is defined by $\tau(t) = t$ so that the trappified canvas accepts whenever the measurement outcome of the central qubit corresponds to the expected 0 outcome. Here, the 3×3 partial pattern defines a trap(see Figure 3).

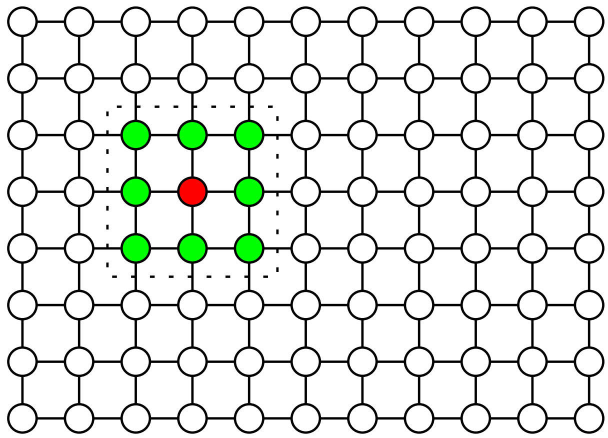


Fig. 3: Trappified canvas. The partial pattern inside the dashed box is the trap. The central qubit (red) is prepared in $|+\rangle$ and is surrounded by $|0\rangle$'s (green) that effectively ensure that irrespective of the measurement angles on the remaining qubits the central qubit will remain in $|+\rangle$. Failure to obtain the 0-outcome when measuring X will be a proof that the server deviated from the given instructions. The preparation and measurement angles of the remaining qubits is left unspecified.

B.3 Embedding Algorithm on a 2D-Cluster Graph Canvas with a Single Trap

Define $\mathfrak{C}$ as the class of computations that can be implemented using a $(n-3)\times m$ 2D-cluster state. An embedding algorithm for $\mathfrak{C}$ on T can be defined in the following way. Consider the trappified canvas T of Example B.2 with a $n \times m$ 2D-cluster graph and a single 3×3 trap in the upper left corner. The output of the embedding algorithm would be the pattern P defined in the following way. For $C \in \mathfrak{C}$, by assumption, one can define a pattern Q on a $(n-3) \times m$ 2D-cluster graph that implements C. The angles and flow of the partial pattern P is identical to that of Q albeit applied on the lower $n-3$ rows of T. On the $3 \times (m-3)$ upper right rectangular subgraph, all angles are set randomly. I_C is such that it comprises all inputs defined in Q and the last $m-3$ qubits of the third row. Choose $\Pi_{I,P}$ so that these $m-3$ qubits are set to $|0\rangle$. Then, by construction, this together with the trap isolates a $(n-3) \times m$ rectangular subgraph on which P will be defining MBQC instructions identical to those of Q, thereby implementing C. In addition, one can see that there are no dependency between measurements of P and that of the trap in T so that the embedding algorithm is proper. Note that one can change the location of the trap to any column. If in addition the 2D-cluster graph if cylindrical instead of rectangular, the trap can be moved to any location within the cylinder.

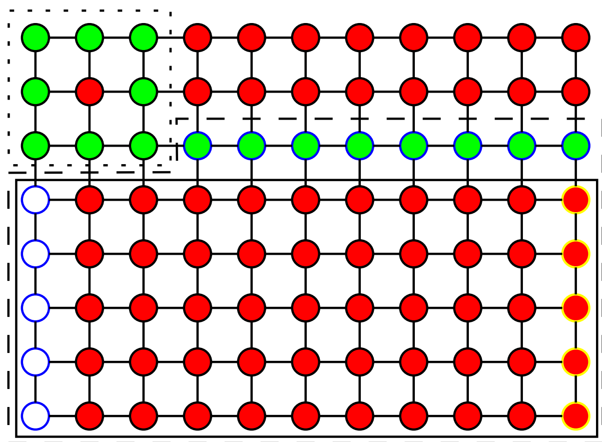


Fig. 4: Trappified canvas. Input qubits are surrounded in blue, output qubits in yellow. Red filled qubits are prepared in $|+\rangle$ while the green ones are prepared in $|0\rangle$, white ones are left unspecified. The trap is located in the upper left corner. The actual computation takes place in the 5×11 rectangular cluster state surrounded by a solid-line while the computation pattern comprises the qubits surrounded by long-dashed line. This allows to include some dummy $|0\rangle$ qubits in the inputs so as to disentangle the lower 5 rows from the rest of the graph and perform the computation. Output qubits of the computation are surrounded by a yellow line. The partial pattern inside the dashed box is the trap. The central qubit (red) is prepared in $|+\rangle$ and is surrounded by $|0\rangle$'s (green) that effectively ensure that irrespective of the measurement angles on the remaining qubits the central qubit will remain in $|+\rangle$. Failure to obtain the 0-outcome when measuring X will be a proof that the server deviated from the given instructions. The preparation and measurement angles of the remaining qubits is left unspecified.

B.4 Trappified Scheme for a Cylindrical-Cluster Graph

Consider the set of trappified canvases together with the embedding algorithm $E_{\mathfrak{C}}$ on the cylindrical cluster-graph with a single randomly placed 3×3 trap as defined in Example B.3. This defines a trapification scheme for $\mathfrak{C}$ consisting of computations that can be implemented using a $(n-3) \times m$ 2D-cluster graph (See Figure 5).

C Proof of Theorems from Section 4

C.1 Proof of Theorem 6

Theorem 6 (Detection Implies Verifiability). *Let $\boldsymbol{P}$ be a trappified scheme with a proper embedding. Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two sets of Pauli deviations such that $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$, and $\mathsf{I} \in \mathcal{E}_2$. If $\boldsymbol{P}$:*
- *ϵ-detects $\mathcal{E}_1$,*
- *is δ-insensitive to $\mathcal{E}_2$,*
- *is ν-correct on $\mathcal{G}_V \setminus \mathcal{E}_1$,*

for $\epsilon, \delta, \nu > 0$, then the Trappified Delegated Blind Computation Protocol 3 for computing CPTP maps in $\mathfrak{C}$ using $\boldsymbol{P}$ is $\delta + \nu$-correct and $\max(\epsilon, \nu)$-secure in the Abstract Cryptograhy framework, i.e. it $\max(\epsilon, \delta + \nu)$-constructs the Secure Delegated Quantum Computation Resource 1 where the leak is defined as $l_\rho = (\mathfrak{C}, G, \boldsymbol{P}, \preceq_G)$.

Proof of Correctness. We start by analysing the correctness of Protocol 3, i.e. the distance between the real and ideal input/output relation if both parties follow their prescribed operations. Let $\mathsf{C} \in \mathfrak{C}$ be the client's desired computation. Let ρ_C be the Client's input state and $|\psi_C\rangle$ a purification of ρ_C using the distinguisher's register D. Let $C \cup T$ be a trappified pattern obtained from sampling a trappified canvas T from the trappified scheme $\boldsymbol{P}$ using probability distribution $\mathcal{P}$ and embedding computation C into in using the embedding algorithm. We denote $\mathsf{C}(\rho_C) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}|$ and $\mathrm{Tr}_{O_C^c}(C \cup T(\rho_C \otimes \sigma) \otimes |\tau(t)\rangle\langle\tau(t)|)$ the final outputs of the Client in the ideal and real settings, where the trace is over all registers not containing the output of the Client's computation.[10] The distinguishing advantage is defined as:

$$\epsilon_{cor} = \| \mathsf{C} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C|) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}| - \tilde{\mathsf{C}}_{T,\mathsf{I}} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma) \otimes |\tau(t)\rangle\langle\tau(t)| \,\|_{\mathrm{Tr}}, \tag{23}$$

where $\tilde{\mathsf{C}}_{T,\mathsf{I}} = \mathsf{D}_{O,C} \circ \mathrm{Tr}_{O_C^c} \circ (C \cup T)$.

In the honest case, the concrete and ideal settings will output different states only in the case where the protocol wrongly rejects the computation or outputs a wrong result despite the absence of errors.

[10] We use here the notation $P(\rho)$ to mean the application of the trappified pattern P to the input state ρ. Also we consider here that the decision function τ outputs either Acc for acceptance or Rej for rejection instead of a binary value.

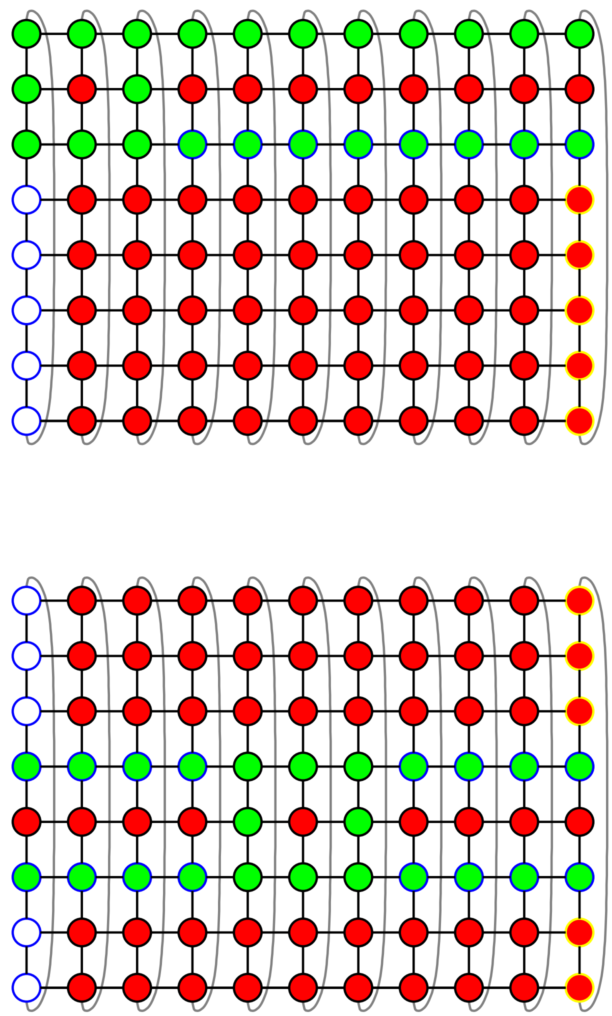


Fig. 5: Trappified scheme. Two possible canvas extracted from a trappified scheme with a single 3×3 trap on a toric 8×11 toric cluster state. Input qubits are surrounded in blue, output qubits in yellow. Red filled qubits are prepared in $|+\rangle$ while the green ones are prepared in $|0\rangle$, white ones are left unspecified. The actual computation takes place in the 5×11 rectangular cluster state, while the trap is located at a different positions in each picture allowing to detect all possible deviations performed by a server, albeit with a low probability of success.

Since the trappified scheme is δ-insensitive to $\mathsf{I} \in \mathcal{E}_2$, the probability that the decision function outputs Rej is bounded by δ as per Definition 10. Furthermore, using Lemma 1, the output of the test is independent of the computation being performed. Combining these two properties yields:

$$\begin{aligned}\epsilon_{cor} \leq \|&\mathsf{C} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C|) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}| - \\ &\tilde{\mathsf{C}}_{T,\mathsf{I}} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma)) \otimes (\delta\,|\mathsf{Rej}\rangle\langle\mathsf{Rej}| + (1-\delta)\,|\mathsf{Acc}\rangle\langle\mathsf{Acc}|)\|_{\mathrm{Tr}}.\end{aligned} \tag{24}$$

Using the convexity of the trace distance, we get:

$$\epsilon_{cor} \leq (1-\delta)\|\mathsf{C} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C|) - \tilde{\mathsf{C}}_{T,\mathsf{I}} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma))\|_{\mathrm{Tr}} + \delta. \tag{25}$$

Finally, the trappified pattern is ν-correct on $\mathsf{I} \in \mathcal{E}_2 \subseteq \mathcal{G}_V \setminus \mathcal{E}_1$. Therefore we have that $\|\mathsf{C} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C|) - \tilde{\mathsf{C}}_{T,\mathsf{I}} \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma))\|_{\mathrm{Tr}} \leq \nu$, meaning that $\epsilon_{cor} \leq (1-\delta)\nu + \delta$. Hence, the protocol is $(\delta + \nu)$-correct. ∎

Proof of Security against Malicious Server. To prove the security of the protocol, we define as per Definition 14 a Simulator σ that has access to the Server's interface of the Secure Delegated Quantum Computation Resource. The interaction involving either the Simulator or the real honest Client should be indistinguishable.

Defining the Server's Simulator. To do so, we use again the fact that when the protocol is run and a deviation is applied by the Server, the probability of accepting or rejecting the computation is dependent only on the deviation and not of the computation performed on the non-trap part of the pattern. This is a crucial property as this allows to simulate the behaviour of the concrete protocol even when the computation performed is unknown. More precisely, we define the Simulator in the following way:

Simulator 1

1. The Simulator request a leak from the Secure Delegated Quantum Computation Resource and receives in return $(\mathfrak{C}, G, \boldsymbol{P}, \preceq_G)$.
2. It chooses at random any computation $\mathsf{C}_S \in \mathfrak{C}$ and an input which is compatible with C.
3. It performs the same tasks as those described by the Client's side of the Trappified Delegated Blind Computation Protocol 3.
4. Whenever τ accepts, the Simulator sends $c = 0$ to the Secure Delegated Quantum Computation Resource, indicating that the honest Client should receive its output. If it rejects, the Simulator sends $c = 1$ Secure Delegated Quantum Computation Resource, indicating an abort.

We now show that the distinguisher cannot tell apart with high probability the simulation and the concrete protocol.

Applying the Pauli Twirl. We first use the twirling lemma to decompose the deviation of the Server. Here we are only concerned with the state representing the interaction of the Client or of the Simulator with the Server. Since the Simulator defined above performs the same tasks as the Client when the Protocol is run, we only need to derive the expression for the Client's interaction. The following steps are similar to the ones in [9, Proof of Theorem 3] and work as can be seen here for the basic UBQC protocol and any protocol based on it.

Let C and ρ_C be the Client's computation and input, let T and σ be the trappified canvas chosen from the trappified scheme $\boldsymbol{P}$ and the associated input. Finally, let $C \cup T$ the trappified pattern resulting from embedding C into T, with base angles $\{\phi(i)\}_{i \in O^c}$.

We start by expressing the state in the simulation and the real protocol. The Server first receives quantum states which are encrypted with $\mathsf{Z}(\theta(i))\mathsf{X}^{a(i)}$ for all vertices $v \in V$. This is explicitly the case for the inputs to the computation and trap patterns, but also for the other qubits of the graph, since we have that $|+_\theta\rangle = \mathsf{Z}(\theta)\,|+\rangle = \mathsf{Z}(\theta)\mathsf{X}^a\,|+\rangle$.[11] Recall that $a_N(i) = \sum_{j \in N_G(i)} a(j)$ and the outputs qubits are only Quantum One-Time-Padded, i.e. $\theta(i) = (r(i) + a_N(v))\pi$ for $i \in O$. Then, omitting the Client's classical registers containing the secret values $\boldsymbol{\theta}, \boldsymbol{a}, \boldsymbol{r}$, state from the point of view of the Client is noted $\rho_{in,\boldsymbol{b}+\boldsymbol{r}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}$, defined as:

$$\rho_{in,\boldsymbol{b}+\boldsymbol{r}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}} = \left(\bigotimes_{i \in V} \mathsf{Z}_i(\theta(i))\mathsf{X}_i^{a(i)}\right)\left(\rho_C \otimes \sigma \otimes |+\rangle\langle+|^{\otimes|V|-|I|}\right) \bigotimes_{i \in O^c} |\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)\rangle\langle\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)|\,, \tag{26}$$

where $\boldsymbol{b}$ corresponds to the perceived branch of computation based on the outcomes returned by the Server to the Client. The values $\delta_{\boldsymbol{b}+\boldsymbol{r}}(i) = (-1)^{a(i)}\phi'_{\boldsymbol{b}+\boldsymbol{r}}(i) + \theta(i) + (r(i) + a_N(i))\pi$ are each encoded as computational basis states on three qubits from a register R with $3n$ qubits. The angle $\phi'_{\boldsymbol{b}+\boldsymbol{r}}(i)$ is obtained through the formula for $\phi'(i)$ from the UBQC Protocol 2, Equation 2 and includes the corrections stemming from $\boldsymbol{b}$ and $\boldsymbol{r}$, while $a_N(i)$ compensates the effect of the X encryption from a qubit on its neighbours. While this seems that the Client is sending the values of $\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)$ at the beginning breaks the causal structure of the protocol, these states will indeed not be affected by any operations before they can actually be correctly computed by the Client. This will be made formal below. Finally, note that for simplicity, the qubits in the state above are not grouped in the order in which the Client sends.

We consider a purification $|\psi_S\rangle\langle\psi_S|$ of the Server's input ρ_S. Let F_{in} be a unitary such that $|\psi_S\rangle = \mathsf{F}_{in}\,|0\rangle^{\otimes w}$ for the appropriate work register size w. Then the operations which the Server applies before any measurement can be written as unitaries acting on all qubits which have not yet been measured and the available values of $\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)$. These can be then decomposed into the correct unitary operation followed by a unitary attack of the Server's choice. The Server receives all qubits, applies the entanglement operation corresponding to the Client's desired graph, then a unitary attack F_G, then the correct rotation on the first measured qubits, followed by another attack F_1. These two last steps are repeated for all measured qubits.

[11] In the real protocol, this value is always 0. This is perfectly indistinguishable since the distribution of the values of δ are identical regardless of this choice of parameter for non-input qubits and correctness is unaffected.

The entanglement according to the graph $G = (V, E)$ is noted $\mathsf{G} = \bigotimes_{(i,j)\in E} \mathsf{CZ}_{i,j}$. The Z-axis rotations required for performing the measurement in the basis defined by $\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)$ are represented by unitaries CR_v, controlled rotations around the Z-axis with the control being performed by the registers containing the corresponding value of $\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)$. Figure 6 shows one possible implementation of this controlled operation.

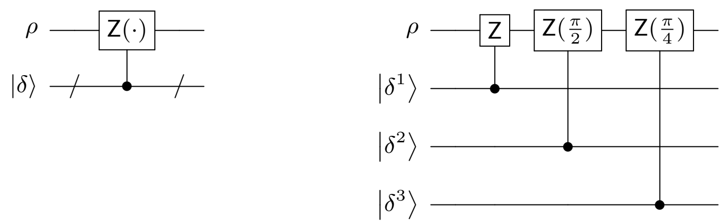


Fig. 6: Controlled rotation used to unitarise Protocol 2. The right hand side is a possible implementation of the rotation on the left, where δ^i are the bits composing the value δ. The 3 controlling qubits are sent by the client to the server in the computational basis as they correspond to classical values.

The quantum state representing the interaction between the Client and Server implementing the protocol just before the measurements are performed, noted $\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{pre,\boldsymbol{b}+\boldsymbol{r}}$, is thus:

$$\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{pre,\boldsymbol{b}+\boldsymbol{r}} = \mathsf{F}_n \mathsf{CR}_n^\dagger \ldots \mathsf{F}_1 \mathsf{CR}_1^\dagger \mathsf{F}_G \mathsf{F}_{in} \mathsf{G}(\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{in,\boldsymbol{b}+\boldsymbol{r}} \otimes |0\rangle\langle 0|^{\otimes w}), \tag{27}$$

with $|O^c| = n$. We can move all deviations through the controlled rotations and regroup them as F'.[12] Then, it is possible to replace the (classically) controlled rotations corresponding to the honest execution of the protocol by ordinary rotations $\mathsf{Z}(\delta_{\boldsymbol{b}+\boldsymbol{r}}(i))^\dagger$, thus yielding:[13]

$$\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{pre,\boldsymbol{b}+\boldsymbol{r}} = \mathsf{F}' \left(\bigotimes_{i\in O^c} \mathsf{Z}_i(\delta_{\boldsymbol{b}+\boldsymbol{r}}(i))^\dagger \right) \mathsf{G}(\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{in,\boldsymbol{b}} \otimes |0\rangle\langle 0|^{\otimes w}). \tag{28}$$

We now apply the decryption operations performed by the Client on the output layer qubits after the Server has returned these qubits at the end of the protocol. The resulting state, noted $\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{dec,\boldsymbol{b}+\boldsymbol{r}}$, can be written as:

$$\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{dec,\boldsymbol{b}+\boldsymbol{r}} = \left(\bigotimes_{i\in O} \mathsf{Z}_i^{s_Z(i)+r(i)} \mathsf{X}_i^{s_X(i)+a(i)} \right) (\rho^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}_{pre,\boldsymbol{b}+\boldsymbol{r}}), \tag{29}$$

where $s_X(i)$ and $s_Z(i)$ stem from the flow of the trappified pattern $C \cup T$. To finish, we enforce that the computation branch is effectively $\boldsymbol{b}$ by projecting

[12] Formally, we have $\mathsf{F}' = \mathsf{F}_n \mathsf{CR}_n^\dagger \ldots \mathsf{F}_1 \mathsf{CR}_1^\dagger \mathsf{F}_G \mathsf{F}_{in} \circ \bigotimes_{i\in O^c} \mathsf{CR}_i$.

[13] If these operations were replaced before, the deviations would pick up a dependency on $\delta_{\boldsymbol{b}+\boldsymbol{r}}(i)$ during the commutation.

all non-output qubits $i \in O^c$ onto $\mathsf{Z}_i^{b(i)+r(i)} |+\rangle\langle+| \mathsf{Z}_i^{b(i)}$.[14,15] Since $|+\rangle$ is a $+1$ eigenstate of X, this is equivalent to projecting onto $\mathsf{Z}_i^{b(i)+r(i)} |+\rangle\langle+|_i \mathsf{X}_i^{a(i)} \mathsf{Z}^{b(i)}$. The final state, noted $\rho_{out,\boldsymbol{b}+\boldsymbol{r}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}$, is therefore:

$$\rho_{out,\boldsymbol{b}+\boldsymbol{r}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}} = \left(\bigotimes_{i\in O^c} \mathsf{Z}_i^{b(i)+r(i)} |+\rangle\langle+|_i\right)\left(\bigotimes_{i\in O^c} \mathsf{X}_i^{a(i)} \mathsf{Z}_i^{b(i)}\right)(\rho_{dec,\boldsymbol{b}+\boldsymbol{r}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}). \tag{30}$$

We then apply the change of variable $b'(i) = b(i) + r(i)$ and then relabelling $b'(i)$ into $b(i)$. This has the effect of removing the influence of $r(i)$ in the corrected measurement angles, transforming $\phi'_{\boldsymbol{b}+\boldsymbol{r}}(i)$ into $\phi'_{\boldsymbol{b}}(i)$:[16]

$$\rho_{out,\boldsymbol{b}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}} = \left(\bigotimes_{i\in O^c} \mathsf{Z}_i^{b(i)} |+\rangle\langle+|_i\right)\left(\bigotimes_{i\in O^c} \mathsf{X}_i^{a(i)} \mathsf{Z}_i^{b(i)+r(i)}\right)(\rho_{dec,\boldsymbol{b}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}) \tag{31}$$

$$= \mathsf{P}_{\boldsymbol{b}} \tilde{\mathsf{U}}_P \left(\rho_C \otimes \sigma \otimes |+\rangle\langle+|^{\otimes|V|-|I|} \bigotimes_{i\in O^c} |\delta_{\boldsymbol{b}}(i)\rangle\langle\delta_{\boldsymbol{b}}(i)| \otimes |0\rangle\langle 0|^{\otimes w}\right). \tag{32}$$

where we defined $\mathsf{P}_{\boldsymbol{b}}$ as:

$$\mathsf{P}_{\boldsymbol{b}} = \bigotimes_{i\in O^c} \mathsf{Z}_i^{b(i)} |+\rangle\langle+|_i\,, \tag{33}$$

and $\tilde{\mathsf{U}}_P$ as:

$$\begin{aligned} \tilde{\mathsf{U}}_P = &\left(\bigotimes_{i\in O^c} \mathsf{X}_i^{a(i)} \mathsf{Z}_i^{b(i)+r(i)}\right)\left(\bigotimes_{i\in O} \mathsf{Z}_i^{s_Z(i)+r(i)} \mathsf{X}_i^{s_X(i)+a(i)}\right) \mathsf{F}' \circ \\ &\left(\bigotimes_{i\in O^c} \mathsf{Z}_i(\delta_{\boldsymbol{b}}(i))^\dagger\right) \mathsf{G} \left(\bigotimes_{i\in V} \mathsf{Z}_i(\theta(i)) \mathsf{X}_i^{a(i)}\right). \end{aligned} \tag{34}$$

We now look at the state from the point of view of the Server, noted $\rho_{out,\boldsymbol{b}}$, which can be written as follows considering that in this case the secret parameters are unknown:

$$\rho_{out,\boldsymbol{b}} = \frac{1}{8^{|O^c|} \cdot 4^{|V|}} \sum_{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}} \rho_{out,\boldsymbol{b}}^{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{r}}. \tag{35}$$

We focus on the state before the projection $\mathsf{P}_{\boldsymbol{b}}$ is applied. The goal is to remove dependencies on $r(i), a(i)$ which appear outside the encryption and decryption

[14] These qubits can be assumed to be measured without loss of generality since (i) the Server needs to produce the values $\boldsymbol{b}$ using its internal state and the values received from the Client and (ii) the operation F' is fully general, meaning that the Server can use it to reorder the qubits before the measurement if it so desires.

[15] The difference in coefficients takes into account the corrections which the Client applies to the outputs of the measurements to account for $r(i)$.

[16] This value uses the formula for $\phi'(i)$ from the Delegated MBQC Protocol 4, Equation 13.

procedures in order to be able to use the twirling lemma, using the fact that these parameters are chosen at random.[17] To this end we cancel out the values of $\theta(i)$ coming from the initial encryption with those which appear in the rotations by $\delta_{\boldsymbol{b}}(i) = (-1)^{a(i)}\phi'_{\boldsymbol{b}}(i) + \theta(i) + (r(i) + a_N(i))\pi$ for $i \in O^c$:

$$\mathsf{Z}_i(\delta_{\boldsymbol{b}}(i))^\dagger \mathsf{G}\mathsf{Z}_i(\theta(i))\mathsf{X}_i^{a(i)} = \mathsf{Z}_i((-1)^{a(i)}\phi'_{\boldsymbol{b}}(i) + (r(i) + a_N(i))\pi)^\dagger \mathsf{G}\mathsf{X}_i^{a(i)}, \quad (36)$$

due to the fact that the entanglement operation consists of CZ operations through which the Z rotations commute. Now, the values $\theta(i)$ appear only through the definition of the angles $\delta_{\boldsymbol{b}}(i)$. Hence, they perfectly One-Time-Pad these angles and summing over $\theta(i)$ yields the perfectly mixed state in the register R. Formally:

$$\rho_{out,\boldsymbol{b}} = \frac{1}{4^{|V|}} \sum_{\boldsymbol{a},\boldsymbol{r}} \mathsf{P}_{\boldsymbol{b}} \tilde{\mathsf{U}}_P \left(\rho_C \otimes \sigma \otimes |+\rangle\langle+|^{\otimes |V|-|I|} \otimes \mathbb{1}_{3n} \otimes |0\rangle\langle 0|^{\otimes w} \right), \quad (37)$$

where $\mathbb{1}_{3n}$ is the perfectly mixed state over the $3n$ qubits of R. This register has thus no effect on either the computation or the traps and is in tensor product with the rest of the state, it can therefore be traced out by assuming without loss of generality that the Server's deviation has no effect on it.

We can now commute the encryption on both sides of the deviation so that the deviation is exactly sandwiched between two identical random Pauli operations. We start on the right side of F' in the expression of $\tilde{\mathsf{U}}_P$. For all qubits in the graph, we need to commute $\mathsf{X}^{a(i)}$ through the entanglement operation first. Since $\mathsf{CZ}_{i,j}\mathsf{X}_i = \mathsf{Z}_j\mathsf{X}_i\mathsf{CZ}_{i,j}$ (and similarly for X_j), using $a_N(i) = \sum_{j \in N_G(i)} a(j)$ we get that:

$$\mathsf{G} \left(\bigotimes_{i \in V} \mathsf{X}_i^{a(i)} \right) = \left(\bigotimes_{i \in V} \mathsf{Z}_i^{a_N(i)} \mathsf{X}_i^{a(i)} \right) \mathsf{G}. \quad (38)$$

The additional $\mathsf{Z}_i^{r(i)+a_N(i)}$ encryption of the output qubits commute unchanged through the entanglement operation G. These encryptions now need to be commuted through the Z rotations for measured qubits:

$$\mathsf{Z}_i((-1)^{a(i)}\phi'_{\boldsymbol{b}}(i) + (r(i) + a_N(i))\pi)^\dagger \mathsf{Z}_i^{a_N(i)}\mathsf{X}_i^{a(i)} = \mathsf{Z}_i^{r(i)}\mathsf{X}_i^{a(i)}\mathsf{Z}_i(\phi'_{\boldsymbol{b}}(i))^\dagger \quad (39)$$

On the other hand, $\mathsf{Z}_i^{a_N(i)}\mathsf{Z}_i^{r(i)+a_N(i)}\mathsf{X}_i^{a(i)} = \mathsf{Z}_i^{r(i)}\mathsf{X}_i^{a(i)}$ is applied on the output qubits. In total, we have that:

$$\left(\bigotimes_{i \in O^c} \mathsf{Z}_i(\delta_{\boldsymbol{b}}(i))^\dagger \right) \mathsf{G} \left(\bigotimes_{i \in V} \mathsf{Z}_i(\theta(i))\mathsf{X}_i^{a(i)} \right) = \mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} \left(\bigotimes_{i \in O^c} \mathsf{Z}_i(\phi'_{\boldsymbol{b}}(i))^\dagger \right) \mathsf{G} \quad (40)$$

where $\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} = \bigotimes_{i \in V} \mathsf{Z}_i^{r(i)}\mathsf{X}_i^{a(i)}$. On the other side of F' in the expression of $\tilde{\mathsf{U}}_P$, we simply have that:

$$\bigotimes_{i \in O^c} \mathsf{X}_i^{a(i)}\mathsf{Z}_i^{b(i)+r(i)} \bigotimes_{i \in O} \mathsf{Z}_i^{s_Z(i)+r(i)}\mathsf{X}_i^{s_X(i)+a(i)} = \left(\bigotimes_{i \in O^c} \mathsf{Z}_i^{b(i)} \bigotimes_{i \in O} \mathsf{Z}_i^{s_Z(i)}\mathsf{X}_i^{s_X(i)} \right) \mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}^\dagger, \quad (41)$$

---

[17] These paramters must be perfectly random as using them multiple times might introduce correlations which the Server can exploit to derandomise the Pauli twirl.

up to a global phase.

We note $\rho_{cor,\boldsymbol{b}} = \left(\bigotimes_{i\in O^c} \mathsf{Z}_i(\phi'_{\boldsymbol{b}}(i))^\dagger\right) \mathsf{G}(\rho_C \otimes \sigma \otimes |+\rangle\langle+|^{\otimes|V|-|I|})$ the correct state before the encryption-deviation-decryption, and define the measurement outcome and final MBQC correction operator $\mathsf{D}_{\boldsymbol{b}} = \bigotimes_{i\in O^c} \mathsf{Z}_i^{b(i)} \bigotimes_{i\in O} \mathsf{Z}_i^{s_Z(i)}\mathsf{X}_i^{s_X(i)}$.

We then obtain:

$$\rho_{out,\boldsymbol{b}} = \frac{1}{4^{|V|}}\mathsf{P}_{\boldsymbol{b}}\mathsf{D}_{\boldsymbol{b}} \sum_{\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\in\mathcal{G}_V} (\mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}} \otimes \mathsf{I}_w)\mathsf{F}'(\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} \otimes \mathsf{I}_w)(\rho_{cor,\boldsymbol{b}} \otimes |0\rangle\langle 0|^{\otimes w}). \tag{42}$$

Without loss of generality we can decompose the Server's deviation in the Pauli operator basis over the graph's vertices as $\mathsf{F}' = \sum_{\mathsf{E}\in\mathcal{G}_V} \alpha_E \mathsf{E} \otimes \mathsf{U}_\mathsf{E}$. Applying the notation $\mathsf{U}(\rho) = \mathsf{U}\rho\mathsf{U}^\dagger$, we get:

$$\rho_{out,\boldsymbol{b}} = \frac{1}{4^{|V|}}\mathsf{P}_{\boldsymbol{b}}\mathsf{D}_{\boldsymbol{b}} \sum_{\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\in\mathcal{G}_V} \mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{F}'\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\rho_{cor,\boldsymbol{b}} \otimes |0\rangle\langle 0|^{\otimes w}\, \mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{F}'^\dagger\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} \tag{43}$$

$$= \frac{1}{4^{|V|}}\mathsf{P}_{\boldsymbol{b}}\mathsf{D}_{\boldsymbol{b}} \sum_{\mathsf{E},\mathsf{E}'\in\mathcal{G}_V} \alpha_\mathsf{E}\alpha^*_{\mathsf{E}'} \sum_{\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\in\mathcal{G}_V} \mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{E}\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\rho_{cor,\boldsymbol{b}}\mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{E}'^\dagger\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} \otimes \mathsf{U}_\mathsf{E}\, |0\rangle\langle 0|^{\otimes w}\, \mathsf{U}^\dagger_{\mathsf{E}'}, \tag{44}$$

where $\alpha^*_{\mathsf{E}'}$ is the complex conjugate of $\alpha_{\mathsf{E}'}$. We now apply Lemma 7, leading to $\sum_{\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\in\mathcal{G}_V} \mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{E}\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\rho_{cor,\boldsymbol{b}}\mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{E}'^\dagger\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} = 0$ for $\mathsf{E} \neq \mathsf{E}'$. Therefore:

$$\rho_{out,\boldsymbol{b}} = \frac{1}{4^{|V|}}\mathsf{P}_{\boldsymbol{b}}\mathsf{D}_{\boldsymbol{b}} \sum_{\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}},\mathsf{E}\in\mathcal{G}_V} |\alpha_\mathsf{E}|^2\mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{E}\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}\rho_{cor,\boldsymbol{b}}\mathsf{Q}^\dagger_{\boldsymbol{a},\boldsymbol{r}}\mathsf{E}^\dagger\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}} \otimes \mathsf{U}_\mathsf{E}\, |0\rangle\langle 0|^{\otimes w}\, \mathsf{U}^\dagger_\mathsf{E}, \tag{45}$$

The result is a CPTP map defined by $\{\mathsf{E} \otimes \mathsf{U}_\mathsf{E}, p_\mathsf{E} = |\alpha_\mathsf{E}|^2\}_{\mathsf{E}\in\mathcal{G}_V}$, a convex combination of Paulis on the graph's vertices tensored with an operation on the Server's internal register. Overall, this shows that the effect of the Server's deviation is – when averaged over the choice of secret parameters – a probabilistic mixture of Pauli operators on the qubits of the graph.

The Pauli encryption and decryption $\mathsf{Q}_{\boldsymbol{a},\boldsymbol{r}}$ commutes up to a global phase with the Pauli deviation E. We can therefore rewrite the state as:

$$\rho_{out,\boldsymbol{b}} = \mathsf{P}_{\boldsymbol{b}}\mathsf{D}_{\boldsymbol{b}} \sum_{\mathsf{E}\in\mathcal{G}_V} p_\mathsf{E}\mathsf{E}(\rho_{cor,\boldsymbol{b}}) \otimes \mathsf{U}_\mathsf{E}(|0\rangle\langle 0|^{\otimes w}) \tag{46}$$

Since the distinguisher wishes to maximise its distinguishing probability, it is sufficient to consider that it applies a fixed Pauli deviation $\mathsf{E} \in \mathcal{G}_V$ for which the distinguishing probability is maximal. Furthermore, the state in the Server's register is unentangled from the rest and therefore does not contribute to the attack of the Server on the Client's state. Once this is traced out, seeing as D_b and E are Paulis and therefore commute up to a global phase, the final state can be written as:

$$\rho_{out,\boldsymbol{b}} = \mathsf{P}_{\boldsymbol{b}}\mathsf{E}\mathsf{D}_{\boldsymbol{b}}(\rho_{cor,\boldsymbol{b}}) = \mathsf{E} \circ (C \cup T)(\rho_C \otimes \sigma). \tag{47}$$

The final equality stems from the definition of the notation $\mathsf{E} \circ P$ for a pattern P (Section 3.1) and the fact that applying $\mathsf{D}_{\boldsymbol{b}}$ to $\rho_{cor,\boldsymbol{b}}$ performs exactly the correct unitary portion of plain MBQC pattern $C \cup T$ – up to the measurements.[18]

Applying the Composable Security of UBQC. We next show that this deviation depends on the same classical parameters in the ideal and real scenarii. To this end, we apply the composition Theorem 11 of the AC framework to replace the execution of the UBQC Protocol by the Blind Delegated Quantum Computation Resource 2 both in the simulation and the real protocol. As per the security of the UBQC Protocol as expressed in Theorem 12, the distinguishing advantage is not modified by this substitution so long as the graph, order of measurements and output set of qubits are known to the Server. The results can be seen in Figures 7 and 8. The distinguisher has access to all outward interfaces.

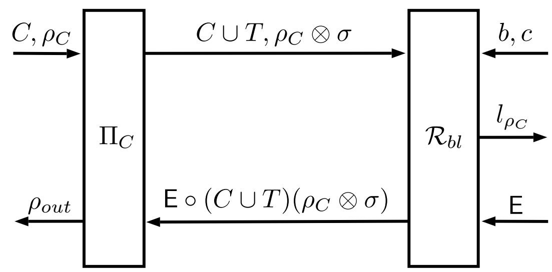


Fig. 7: Real world hybrid interaction between the Client's protocol CPTP map Π_C and Blind Delegated QC Resource $\mathcal{R}_{bl}$.

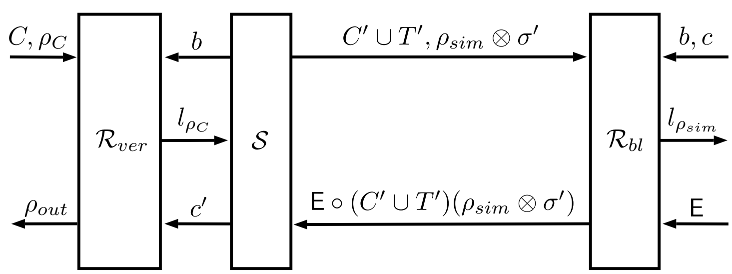


Fig. 8: Simulator $\mathcal{S}$ interacting Secure and Blind DQC Resources $\mathcal{R}_{ver}$ and $\mathcal{R}_{bl}$.

The Simulator receives the leak $l_{\rho_C} = (\mathfrak{C}, G, \boldsymbol{P}, \preceq_G)$ from the Secure Delegated Quantum Computation Resource. In both cases, we assume that both the Client and Simulator send $(\mathfrak{C}, G, \boldsymbol{P}, \preceq_G)$ as a first message to the Server. All canvases in $\boldsymbol{P}$ are blind-compatible (Definition 7) meaning that they all share the graph G and the same output set O, and the order $\preceq_G$ is used for all patterns generated from $\boldsymbol{P}$ and the embedding algorithm. Since these parameters are the same in all

[18] This is correct up to a relabelling of $\mathcal{G}_V$ since in the rest of the paper we assumed that the measurements are performed in the computational basis.

executions of both the real and ideal case, the leak $l_{\rho_{ideal}}$ obtained by the Server in the simulation does not yield any more information. Overall, the classical information in both cases is identical and does not help the distinguisher on its own.

Output and Abort Probability Analysis. The interactions are therefore indistinguishable before the output is sent back to the Client and we focus in the following on the output state and the abort probability in both cases, which are the only remaining elements which the distinguisher can use to decide which setup it is interacting with. The outputs $\rho_{\substack{out\\real}}, \rho_{\substack{out\\ideal}}$ of the Client in respectively the real and ideal settings can be written as follows for Pauli deviation $\mathsf{E} \in \mathcal{G}_V$ introduced by the distinguisher:

$$\rho_{\substack{out\\real}} = p_{\substack{\mathsf{Acc}\\real}} \tilde{\mathsf{C}}_{T,\mathsf{E}}(\rho_C \otimes \sigma) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}| + \left(1 - p_{\substack{\mathsf{Acc}\\real}}\right) |\bot\rangle\langle\bot| \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|, \quad (48)$$

$$\rho_{\substack{out\\ideal}} = p_{\substack{\mathsf{Acc}\\ideal}} \mathsf{C}(\rho_C) \otimes |\mathsf{Acc}\rangle\langle\mathsf{Acc}| + \left(1 - p_{\substack{\mathsf{Acc}\\ideal}}\right) |\bot\rangle\langle\bot| \otimes |\mathsf{Rej}\rangle\langle\mathsf{Rej}|, \quad (49)$$

where $\tilde{\mathsf{C}}_{T,\mathsf{E}} = \mathsf{D}_{O,C} \circ \mathrm{Tr}_{O_C^c} \circ \mathsf{E} \circ (C \cup T)$ and for all $\mathsf{C} \in \mathfrak{C}$ we have that:

$$p_{\substack{\mathsf{Acc}\\\mathsf{C}}} = \sum_{T \in \boldsymbol{P}} \Pr_{T \sim \mathcal{P}} \left[\tau\left(\mathrm{Tr}_{O_T^c}\left(\mathsf{E} \circ (C \cup T)(\rho_C \otimes \sigma)\right)\right) = 0, T\right] \quad (50)$$

$$= \sum_{T \in \boldsymbol{P}} \Pr_{\substack{T \sim \mathcal{P}\\ t \sim \mathsf{E} \circ \mathcal{T}}} [\tau(t) = 0, T], \quad (51)$$

which uses Lemma 1, namely that the acceptance probability does not depend on the input or the computation. Therefore this probability is identical in the real and ideal setting $p_{\substack{\mathsf{Acc}\\real}} = p_{\substack{\mathsf{Acc}\\ideal}} = p_{\mathsf{Acc}}$, regardless of the deviation chosen by the distinguisher. We see that whenever the computation is rejected, the output state is identical in both setups. On the other hand, whenever the computation is accepted, the ideal resource will always output the correct state, while the concrete protocol outputs a potentially erroneous state. By convexity of the trace distance, the distinguishing probability p_d can therefore be written as:

$$p_d = \max_{\substack{\mathsf{E} \in \mathcal{G}_V\\ \mathsf{C} \in \mathfrak{C}\\ \psi_C}} \left(p_{\mathsf{Acc}} \times \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C} \otimes \mathsf{I}_T) \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma)\|_{\mathrm{Tr}}\right), \quad (52)$$

where $|\psi_C\rangle$ is a purification of the Client's input ρ_C using the distinguisher's register D. We now therefore analyse the output state in the case where the computation is accepted.

Error Influence on Distinguishing Probability. First consider the case where $\mathsf{E} \in \mathcal{E}_1$. Since $\boldsymbol{P}$ ϵ-detects such errors (Definition 9), the probability of accepting is upper-bounded by ϵ, which implies:

$$p_{d,\mathcal{E}_1} \leq \epsilon \times \max_{\substack{\mathsf{E} \in \mathcal{E}_1\\ \mathsf{C} \in \mathfrak{C}\\ \psi_C}} \left(\|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C} \otimes \mathsf{I}_T) \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma)\|_{\mathrm{Tr}}\right). \quad (53)$$

The distinguisher can freely choose the Client's input state ψ_C and the computation $\mathsf{C} \in \mathfrak{C}$ and there is no constraint on the effect of this deviation on the computation part of the trappified pattern. In the worst case the incorrect real output state is orthogonal to the ideal output state, meaning that the distinguisher can tell apart both settings with certainty and the trace distance is upper-bounded by 1. The distinguishing probability in this scenario therefore follows $p_{d,\mathcal{E}_1} \leq \epsilon$.

Second, we consider the alternate case, where $\mathsf{E} \notin \mathcal{E}_1$. Here, we assumed that the trappified scheme $\boldsymbol{P}$ is ν-correct on the set $\mathcal{G}_V \setminus \mathcal{E}_1$ (Definition 11), therefore the trace distance between the correct result of the computation and the real output of the protocol is upper-bounded by ν:

$$\|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C}) \otimes \mathsf{I}_D(|\psi_C\rangle\langle\psi_C| \otimes \sigma)\|_{\mathrm{Tr}} \leq \nu. \tag{54}$$

Therefore:

$$p_{d,\mathcal{G}_V \setminus \mathcal{E}_1} \leq \nu \times \max_{\mathsf{E} \in \mathcal{G}_V \setminus \mathcal{E}_1}(p_{\mathsf{Acc}}), \tag{55}$$

where the maximisation is done only over the error since the acceptance probability is independent of the input and computation. In this case, the accepting probability p_{Acc} is not constrained and hence only upper bounded by 1, yielding $p_{d,\mathcal{G}_V \setminus \mathcal{E}_1} \leq \nu$.

Since the deviation chosen by the distinguisher falls in either of these two cases, we have $p_d = \max(p_{d,\mathcal{E}_1}, p_{d,\mathcal{G}_V \setminus \mathcal{E}_1})$ and the maximum distinguishing probability between the Resource together with the Simulator and the concrete Protocol is thus upper-bounded by $\max(\epsilon, \nu)$. ■

C.2 Proof of Theorem 7

Theorem 7 (Robust Detection Implies Robust Verifiability). *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two sets of Pauli deviations such that $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$ and $\mathsf{I} \in \mathcal{E}_2$. Let $\boldsymbol{P}$ be a trappified scheme for computation set $\mathfrak{C}$, which is δ-insensitive to $\mathcal{E}_2$ and ν-correct on $\mathcal{G}_V \setminus \mathcal{E}_1$. Let $C \cup T$ be a trappified pattern resulting from embedding computation $\mathsf{C} \in \mathfrak{C}$ in trappified canvas T sampled from $\boldsymbol{P}$. We assume an execution of Protocol 3 with an honest-but-noisy Server whose noise is modelled by sampling an error $\mathsf{E} \in \mathcal{E}_2$ with probability $(1 - p_2)$ and $\mathsf{E} \in \mathcal{G}_V \setminus \mathcal{E}_2$ with probability p_2. Then, the Client in Protocol 3 accepts with probability at least $(1 - p_2)(1 - \delta)$, and if accepted the distance between the implemented transformation and the client's computation is bounded as follows:*

$$\forall \mathsf{C} \in \mathfrak{C},\ \max_{\psi} \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C}) \otimes \mathsf{I}_R(|\psi_C\rangle\langle\psi_C| \otimes \sigma)\|_{\mathrm{Tr}} \leq \nu + p_2 + \delta, \tag{56}$$

where $|\psi_C\rangle$ is a purification of the Client's input ρ_C using auxiliary quantum register R, and $\tilde{\mathsf{C}}_{T,\mathsf{E}} = \mathsf{D}_{O,C} \circ \mathrm{Tr}_{O_C^c} \circ \mathsf{E} \circ (C \cup T)$.

Proof. By construction, $\boldsymbol{P}$ is δ-insensitive to $\mathcal{E}_2$. Hence, it will accept deviations in $\mathcal{E}_2$ with probability at least $1 - \delta$ which yields the overall lower bound on the acceptance probability of $(1 - p_2)(1 - \delta)$.

There are then two cases when the computation is accepted. If the deviation E is in $\mathcal{E}_2 \subseteq \mathcal{G}_V \setminus \mathcal{E}_1$, we have by definition:

$$\forall \mathsf{C} \in \mathfrak{C},\ \max_{\psi} \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C}) \otimes \mathsf{I}_R(|\psi\rangle\langle\psi| \otimes \sigma)\|_{\mathrm{Tr}} \leq \nu. \tag{57}$$

Otherwise, if the deviation is not in $\mathcal{E}_2$ but is accepted, the distance can always be bounded by 1.

The first case happens with probability at least $(1-p_2)(1-\delta)$, since according to Bayes' theorem:

$$\Pr[\mathsf{E} \in \mathcal{E}_2 \mid \mathsf{Acc}] = \frac{\Pr[\mathsf{Acc} \mid \mathsf{E} \in \mathcal{E}_2] \cdot \Pr[\mathsf{E} \in \mathcal{E}_2]}{\Pr[\mathsf{Acc}]} \geq (1-p_2)(1-\delta). \tag{58}$$

Consequently, for the second case it holds then that:

$$\Pr[\mathsf{E} \notin \mathcal{E}_2 \mid \mathsf{Acc}] = 1 - \Pr[\mathsf{E} \in \mathcal{E}_2 \mid \mathsf{Acc}] \leq 1 - (1-p_2)(1-\delta) = p_2 + \delta - p_2\delta. \tag{59}$$

Using the convexity of the trace norm, for the case of an accepting run of the protocol we finally arrive at

$$\forall \mathsf{C} \in \mathfrak{C},\ \max_{\psi} \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C}) \otimes \mathsf{I}_R(|\psi\rangle\langle\psi| \otimes \sigma)\|_{\mathrm{Tr}} \tag{60}$$

$$\leq \nu \Pr[\mathsf{E} \in \mathcal{E}_2 \mid \mathsf{Acc}] + 1 - \Pr[\mathsf{E} \in \mathcal{E}_2 \mid \mathsf{Acc}] = 1 - (1-\nu)\Pr[\mathsf{E} \in \mathcal{E}_2 \mid \mathsf{Acc}] \tag{61}$$

$$\leq 1 - (1-\nu)(1-p_2)(1-\delta) = \nu(1-p_2)(1-\delta) + p_2 + \delta - p_2\delta, \tag{62}$$

which concludes the proof. ■

C.3 Proof of Theorem 8

Theorem 8 (Error-Correction Prevents Resource Blow-up). *Let C be a minimal MBQC pattern implementing a CPTP map C. Let $C \cup T$ denote a trappified pattern implementing C obtained from $\boldsymbol{P}(\lambda)$. Further assume that Protocol 3 using $\boldsymbol{P}(\lambda)$ has negligible security error $\max(\epsilon, \nu)$ with respect to λ.*

If $|O_{C\cup T}|/|O_C| = 1$ for a non-negligible fraction of trappified canvases $T \in \boldsymbol{P}(\lambda)$, then the overhead $|G_{\boldsymbol{P}(\lambda)}|/|G_C|$ is super-polynomial in λ.

Proof. Consider a trappified pattern $C \cup T$ for computing C obtained from $\boldsymbol{P}(\lambda)$ such that $|O_{C\cup T}| = |O_C|$. Given $\preceq_{G_{\boldsymbol{P}(\lambda)}}$, let $o_{C\cup T} \in O_{C\cup T}$ be the first output position of the computation. By definition, a bit-flip operation applied on position $o_{C\cup T}$ cannot be detected by the trap in $C \cup T$ since the outcome of the trap is independent of the computation. Yet, because C is minimal and $|O_{C\cup T}| = |O_C|$, we get that for some input states, the bit-flip deviation on $o_{C\cup T}$ si harmful. As a consequence, there exists a λ_0 such that, for all $\lambda \geq \lambda_0$, the diamond distance between C and the bit-flipped version is greater than $\nu(\lambda)$. To obtain exponential verification it is therefore necessary for this bit flip to be in the set of ϵ-detected deviations. This means that deviating on this position without being detected can happen for at most a negligible fraction $\eta(\lambda)$ of the trappified canvases in $\boldsymbol{P}(\lambda)$. In other words, the position $o_{C\cup T}$ can only be the first output computation qubit for a negligible fraction $\eta(\lambda)$ of trappified patterns in $\boldsymbol{P}(\lambda)$ that satisfy $|O_{C\cup T}| = |O_C|$.

Then define $\tilde{\boldsymbol{P}}(\lambda) = \{P = E_{\mathfrak{C}}(\mathsf{C}, \boldsymbol{P}(\lambda)),\ |O_{C\cup T}| = |O_C|\}$ as the set of trappified patterns for $\mathcal{C}$ that have no overhead, and $O = \{o_{C\cup T},\ T \in \tilde{\boldsymbol{P}}(\lambda)\}$ the set of vertices corresponding to their first output location. By construction, we have $\sum_{o\in O} |\{T \in \tilde{\boldsymbol{P}},\ o_{C\cup T} = o\}| = |\tilde{\boldsymbol{P}}|$. But, we just showed that $|\{T \in \tilde{\boldsymbol{P}},\ o_{C\cup T} = o\}|/|\boldsymbol{P}(\lambda)|$ is upper-bounded by η, negligible in λ. Thus, $|O|$ is lower-bounded by $|\tilde{\boldsymbol{P}}(\lambda)|/(|\boldsymbol{P}(\lambda)|\eta)$ which is super-polynomial in λ so long as $|\tilde{\boldsymbol{P}}(\lambda)|/|\boldsymbol{P}(\lambda)|)$ is not negligible in λ. ■

Note that the situation where $|O_{C\cup T}| > |O_C|$ is interesting only if the bit-flip deviation on qubit $o_{C\cup T}$ does not alter the computation. Otherwise, the same reasoning as above is still applicable. This shows that enlarging the physical Hilbert space storing the output of the computation is useful only if it allows for some error-correction which transforms low-weight harmful errors into harmless ones.

D Proof of Theorems from Section 5

D.1 Proof of Theorem 9

Theorem 9 (From Constant to Exponential Detection and Insensitivity Rates).
Let $\boldsymbol{P}$ be a trappified scheme on graph G which ϵ-detects the error set $\mathcal{E}_1$, is δ-insensitive to $\mathcal{E}_2$ and perfectly insensitive to $\{\mathsf{I}\}$. For $d, s \in \mathbb{N}$, $n = d + s$ and $w \in [s]$, let $\boldsymbol{P}'$ be the trappified scheme resulting from the compilation defined in Definition 12.

For $\mathsf{E} \in \mathcal{G}_{V^n}$, let $\mathrm{wt}(\mathsf{E})$ be defined as the number of copies of G on which E does not act as the identity. We define $\mathcal{E}_{\geq k,\mathcal{F}} = \{\mathsf{E} \in (\mathcal{F} \cup \{\mathsf{I}\})^n \mid \mathrm{wt}(\mathsf{E}) \geq k\}$, and $\mathcal{E}_{\leq k,\mathcal{F}}$ analogously.

Let $k_1 > nw/(s\epsilon)$ and $k_2 < nw/(s\delta)$. Then, $\boldsymbol{P}'$ ϵ'-detects $\mathcal{E}_{\geq k_1,\mathcal{E}_1}$ and is δ'-insensitive to $\mathcal{E}_{\leq k_2,\mathcal{E}_2}$ where:

$$\epsilon' = \min_{\chi\in\left[0,\frac{k_1}{n}-\frac{w}{s\epsilon}\right]} \exp\left(-2\chi^2 s\right) + \exp\left(-2\frac{\left(\left(\frac{k_1}{n}-\chi\right)s\epsilon - w\right)^2}{\left(\frac{k_1}{n}-\chi\right)s}\right), \tag{63}$$

$$\delta' = \min_{\chi\in\left[0,\frac{w}{s\delta}-\frac{k_2}{n}\right]} \exp\left(-2\chi^2 s\right) + \exp\left(-2\frac{\left(\left(\frac{k_2}{n}+\chi\right)s\delta - w\right)^2}{\left(\frac{k_2}{n}+\chi\right)s}\right). \tag{64}$$

Proof. For a given deviation E, let X be a random variable describing the number of test rounds on which the deviation's action is not identity, where the probability is taken over the choice of the trappified canvas $\boldsymbol{P}'$. Let Y be a random variable counting the number of test rounds for which the decision function rejects.

Let $x \in [s]$, we can always decompose $\Pr[Y < w]$ as:

$$\Pr[Y < w] = \Pr[Y < w \mid X \le x] \Pr[X \le x] + \Pr[Y < w \mid X > x] \Pr[X > x] \tag{65}$$

$$\le \Pr[X \le x] + \Pr[Y < w \mid X > x]. \tag{66}$$

We now aim to bound both terms above.

Let $\mathsf{E} \in \mathcal{E}_{\ge k_1, \mathcal{E}_1}$. In this case, by definition of E and construction of $\boldsymbol{P}'$, X is lower-bounded in the usual stochastic order by a variable $\tilde{X}$ following a hypergeometric variable distribution of parameters (n, k_1, s). We fix $x = \left(\frac{k_1}{n} - \chi\right) s$ for $\chi \ge 0$ and use tail bounds for the hypergeometric distribution to get:

$$\Pr\left[X \le \left(\frac{k_1}{n} - \chi\right) s\right] \le \Pr\left[\tilde{X} \le \left(\frac{k_1}{n} - \chi\right) s\right] \le \exp\left(-2\chi^2 s\right). \tag{67}$$

For the other term, note that Y, conditioned on a lower bound x for X, is lower-bounded in the usual stochastic order by an (x, ϵ)-binomially distributed random variable $\tilde{Y}$. Hoeffding's inequality for the binomial distribution then implies that:

$$\Pr[Y < w \mid X > x] \le \Pr\left[\tilde{Y} < w\right] \le \exp\left(-2\frac{(x\epsilon - w)^2}{x}\right). \tag{68}$$

All in all, replacing the value of x above with $\left(\frac{k_1}{n} - \chi\right) s$ and combining it with the first bound, we have for $\chi \le \frac{k_1}{n} - \frac{w}{s\epsilon}$ that:

$$\Pr[Y < w] \le \exp\left(-2\chi^2 s\right) + \exp\left(-2\frac{\left(\left(\frac{k_1}{n} - \chi\right) s\epsilon - w\right)^2}{\left(\frac{k_1}{n} - \chi\right) s}\right). \tag{69}$$

This concludes the first statement.

For the second statement, we can similarly decompose $\Pr[Y \ge w]$ as:

$$\Pr[Y \ge w] \le \Pr[Y \ge w \mid X < x] + \Pr[X \ge x]. \tag{70}$$

Let $\mathsf{E} \in \mathcal{E}_{\le k_2, \mathcal{E}_2}$. Now X is upper-bounded in the usual stochastic order by a variable $\tilde{X}$ following a hypergeometric distribution of parameters (n, k_2, s), by definition of E. This holds here because the scheme is perfectly insensitive to I, and therefore the identity never triggers tests. It then holds for all $\chi \ge 0$ that

$$\Pr\left[X \ge \left(\frac{k_2}{n} + \chi\right) s\right] \le \Pr\left[\tilde{X} \ge \left(\frac{k_2}{n} + \chi\right) s\right] \le \exp\left(-2\chi^2 s\right), \tag{71}$$

using tail bounds for the hypergeometric distribution.

Similarly, here, Y (conditioned on an upper bound x for X) is upper-bounded in the usual stochastic order by an (x, δ)-binomially distributed random variable

$\tilde{Y}$. This also holds because of the perfect insensitivity of tests to I. Hoeffding's inequality yields

$$\Pr\left[Y \geq w \mid X \leq x\right] \leq \Pr\left[\tilde{Y} \geq w\right] \leq \exp\left(-2\frac{(x\delta - w)^2}{x}\right). \tag{72}$$

We then conclude for $\chi \leq \frac{w}{s\delta} - \frac{k_2}{n}$ that

$$\Pr\left[Y \geq w\right] \leq \exp\left(-2\chi^2 s\right) + \exp\left(-2\frac{\left(\left(\frac{k_2}{n} + \chi\right) s\delta - w\right)^2}{\left(\frac{k_2}{n} + \chi\right) s}\right). \qquad \blacksquare$$

D.2 Proof of Theorem 10

Theorem 10 (Exponential Correctness from Majority Vote). *Let $\boldsymbol{T}$ be a trappified scheme on graph G which is perfectly correct on $\{\mathsf{I}\}$, for computations $\mathfrak{C} = \mathsf{BQP} \cap \mathfrak{G}$ where $\mathfrak{G}$ is the set of MBQC computations which can be performed on graph G. For $d, s \in \mathbb{N}$ and $n = d + s$, let $\boldsymbol{V}$ be the trappified scheme obtained through the compiler of Definition 12 and let the input subspace $\Pi_{I,C}$ be symmetric with respect to exchanging computation rounds. The output subspace $\Pi_{O,C}$ is defined as the concatenation of the (classical) outputs of all computation rounds and the decoding algorithm $\mathsf{D}_{O,C}$ is the bitwise majority vote of computation rounds outputs from the d computations.*

Let c be the bounded error of BQP computations and $k < \frac{2c-1}{2c-2}n$. Then, $\boldsymbol{V}$ is ν-correct on $\mathcal{E}_{\leq k,\mathcal{G}_V}$ for

$$\nu \leq \exp\left(-2\left(1 - \frac{2c-1}{2c-2} + \varphi - \epsilon_1\right) d\epsilon_2^2\right), \tag{73}$$

with

$$\frac{1}{2} - \left(\frac{2c-1}{2c-2} - \varphi + \epsilon_1\right) = (c + \epsilon_2)\left(1 - \frac{2c-1}{2c-2} + \varphi - \epsilon_1\right) \tag{74}$$

and $\varphi, \epsilon_1, \epsilon_2 > 0$. Thus ν is exponentially small in n if d/n is constant.

Proof. We will compute the bound on the correctness for finite n. First, define two random variables Z_1 and Z_2 that account for possible sources of erroneous results for individual computation rounds. More precisely, Z_1 is the number of computation rounds that are affected by a deviation containing an Y or Z for one of the qubits in the round. Z_2 is the number of computation rounds which give the wrong outcome due to the probabilistic nature of the computation itself – i.e. inherent failures for the computation in the honest and noise free case. Given that $\boldsymbol{V}$ uses a majority vote to recombine the results of each computation rounds, as long a $Z_1 + Z_2 < d/2$, then the output result will be correct.

Our goal now is to show that the probability that $Z_1 + Z_2$ is greater than $d/2$ can be made negligible. For any z_1 one has the following:

$$\Pr\left[Z_1 + Z_2 \geq \frac{d}{2}\right] = \Pr\left[Z_1 + Z_2 \geq \frac{d}{2} | Z_1 \leq z_1\right] \Pr[Z_1 \leq z_1] \tag{75}$$

$$+ \Pr\left[Z_1 + Z_2 \geq \frac{d}{2} | Z_1 > z_1\right] \Pr[Z_1 > z_1]. \tag{76}$$

Then:

$$\Pr\left[Z_1 + Z_2 \geq \frac{d}{2}\right] \leq \Pr\left[Z_1 + Z_2 \geq \frac{d}{2} | Z_1 \leq z_1\right] + \Pr[Z_1 > z_1] \tag{77}$$

$$\leq \Pr\left[Z_2 \geq \frac{d}{2} - z_1 | Z_1 \leq z_1\right] + \Pr[Z_1 > z_1] \tag{78}$$

$$\leq \Pr\left[Z_2 \geq \frac{d}{2} - z_1 | Z_1 = z_1\right] + \Pr[Z_1 > z_1]. \tag{79}$$

Now, consider a deviation in $\mathcal{E}_{\leq k, \mathcal{G}_V}$. Using the tail bound for the hypergeometric distribution defined by choosing independently at random and without replacement d computation rounds out of a total of n rounds, k of which at most are affected by the deviation, one finds that for $z_1 = (k/n + \epsilon_1)d$ with $0 < \epsilon_1$,

$$\Pr\left[Z_1 > \left(\frac{k}{n} + \epsilon_1\right) d\right] \leq \exp\bigl(-2\epsilon_1^2 d\bigr). \tag{80}$$

Additionally, once Z_1 is fixed, Z_2 is biniomally distributed with probability c. Therefore, using tail bound for this distribution, one has for $\epsilon_2 > 0$:

$$\Pr\left[Z_2 \geq (c + \epsilon_2)\left(1 - \frac{k}{n} - \epsilon_1\right) d | Z_1 = (\frac{k}{n} + \epsilon_1) d\right] \leq \exp\left(-2\left(1 - \frac{k}{n} - \epsilon_1\right) d\epsilon_2^2\right). \tag{81}$$

Using these inequalities, we obtain that:

$$\Pr\left[Z_1 + Z_2 \geq \frac{d}{2}\right] \leq \exp\bigl(-2\epsilon_1^2 d\bigr) + \exp\left(-2\left(1 - \frac{k}{n} - \epsilon_1\right) d\epsilon_2^2\right), \tag{82}$$

where we set

$$\frac{d}{2} - \left(\frac{k}{n} + \epsilon_1\right) d = (c + \epsilon_2)\left(1 - \frac{k}{n} - \epsilon_1\right) d, \tag{83}$$

which has solutions for $\epsilon_1, \epsilon_2 > 0$ when $k/n = (2c-1)/(2c-2) - \varphi$ with $\varphi > 0$. This shows that the correctness error $\nu = \Pr[Z_1 + Z_2 \geq d/2]$ can therefore be made negligible in n for fixed d/n.

■

E Proof of Lemmas from Section 6

Lemma 3 (Tensor Product Preparation of a State in a Stabiliser Subspace). *Let* P *be an element of the Pauli group over* N *qubits, such that* $\mathsf{P}^2 \neq -\mathsf{I}$. *Then, there exists* $|\psi\rangle = \bigotimes_{i=1}^N |\psi_i\rangle$ *such that* $|\psi\rangle = \mathsf{P}|\psi\rangle$, *and* $\forall i, |\psi_i\rangle \in \{|0\rangle, |+\rangle, |+_i\rangle\}$.

Proof. Without loss of generality, one can write $\mathsf{P} = s\bigotimes_i \mathsf{P}(i)$ with $s = \pm 1$ and where $\mathsf{P}(i) \in \{\mathsf{I}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$ is the restriction of P to qubit i. Then by construction, $\mathsf{P} \in \langle S \rangle$, where $\langle \mathcal{S} \rangle$ denotes the multiplicative group generated by the set $\mathcal{S} =$

$\{s\mathsf{P}(i_0)\bigotimes_{j\neq i_0}\mathsf{I}\}\cup\{\mathsf{P}(i)\bigotimes_{j\neq i}\mathsf{I}\}_{i\neq i_0}$, where i_0 is the smallest index i for which $\mathsf{P}(i)\neq\mathsf{I}$. Now, consider the state that is obtained by taking the tensor product of single qubit states that are the common $+1$ eigenstates of the operators in set $\mathcal{S}$. The above shows that it is a $+1$ eigenstate of all operators in $\langle\mathcal{S}\rangle$, and in particular of P, which concludes the proof as eigenstates of single-qubit Pauli operators are precisely the desired set. ■

Lemma 4 (Detection Rate). *Let $G=(V,E)$ be an undirected graph. Let $\mathcal{D}$ be a probability distribution over $\mathcal{I}(G)$, giving rise to the trappified scheme $\boldsymbol{P}$ where every element of $\mathcal{I}(G)$ describes one trappified canvas. We define the* detection rate *of $\mathcal{D}$ over G as $p_{det}(\mathcal{D})=1-\epsilon(\mathcal{D})=\min_{\substack{M\subseteq V\\ M\neq\emptyset}}\Pr_{H\sim\mathcal{D}}[M\cap H\neq\emptyset]$. Then $\boldsymbol{P}$ $\epsilon(\mathcal{D})$-detects the error set $\mathcal{E}=\{\mathsf{I},\mathsf{X},\mathsf{Y},\mathsf{Z}\}^{\otimes V}\setminus\{\mathsf{I},\mathsf{Z}\}^{\otimes V}$.*

Proof. The trappified canvas induced by the independent set $H\in\mathcal{I}(G)$ detects an error E if and only if $M\cap H\neq\emptyset$, where M is the set of all vertices on which E reduces to the Pauli-X or Y. The claim is then implied by Lemma 2. ■

Lemma 5. *For every (non-null) graph G there exists a distribution $\mathcal{D}$ over $\mathcal{I}(G)$ such that $p_{det}(\mathcal{D})\geq\frac{1}{\chi_f(G)}$, with $\chi_f(G)$ the fractional chromatic number of G. Further, for every distribution $\mathcal{D}'$ over $\mathcal{I}(G)$ it holds that $p_{det}(\mathcal{D}')\leq\frac{1}{\omega_f(G)}$, with $\omega_f(G)$ the fractional clique number of G.*

Proof. We use the following characterisation of the detection rate to determine upper bounds on p_{det}: For any graph G and any distribution $\mathcal{D}$ over $\mathcal{I}(G)$ it holds that $p_{\mathrm{det}}(\mathcal{D})=\min_{\mathcal{M}}\Pr_{\substack{M\sim\mathcal{M}\\ H\sim\mathcal{D}}}[M\cap H\neq\emptyset]$, where the minimum ranges over distributions $\mathcal{M}$ over $\wp(V)\setminus\{\emptyset\}$.

Let $\mathcal{D}$ be a distribution over $\mathcal{I}(G)$ such that for all $v\in V$ it holds that $\Pr_{H\sim\mathcal{D}}[v\in H]\geq\frac{1}{k}$. For all $M\subseteq V, M\neq\emptyset$, then $\Pr_{H\sim\mathcal{D}}[M\cap H\neq\emptyset]\geq\frac{1}{k}$ and therefore $p_{\mathrm{det}}(\mathcal{D})\geq\frac{1}{k}$. By Lemma 9, we can find such a distribution $\mathcal{D}$ for any $k\geq\chi_f(G)$, which proves the first claim. The second statement is a direct consequence of Lemma 10. ■

Lemma 6 (General Stabiliser-Based Trappified Scheme). *Let $\boldsymbol{P}$ be the trappified scheme defined by sampling uniformly at random a non-empty set $H\subseteq V$ and preparing the trappified canvas associated to $\mathcal{R}_H=\{\prod_{i\in H}\mathsf{S}_i\}$. Then $\boldsymbol{P}$ $1/2$-detects the error set $\mathcal{E}=\{\mathsf{I},\mathsf{X},\mathsf{Y}\}^{\otimes V}\setminus\{\mathsf{I}^{\otimes V}\}$.*

Proof. Looking at a given deviation E, we conclude that a test-round defined by H detects E if and only if $|E\cap H|$ is odd – here E denotes the set of qubits where E is equal to X or Y. If H is sampled uniformly at random from $\wp(G)$, then $\Pr_{H\sim\mathcal{U}(\wp(G))}[|E\cap H|\equiv 1 \mod 2]=1/2$, and this is valid for any $\mathsf{E}\neq\mathsf{I}$. ■

F General Parallel Repetition

We here show an alternative method for performing the same decomposition, by focusing solely on the error-detection amplification of classical input computations.

We then recover the results above as a consequence of this generic amplification. We start as above by defining a compiler taking as input a trappified scheme and running it several times in parallel before thresholding over the outcomes of the individual decision functions.

Definition 20 (Parallel Repetition Compiler). *Let $(\boldsymbol{P}, \preceq_G, \mathcal{P}, E_{\mathfrak{C}})$ be trappified scheme over a graph G for computation class $\mathfrak{C}$ with classical inputs, and let $n \in \mathbb{N}$ and $w \in [n-1]$. We define the Parallel Repetition Compiler that turns $\boldsymbol{P}$ into a trappified scheme $\boldsymbol{P}_{\|n}$ on G^n for computation class $\mathfrak{C}$ as follows:*

- *The set of trappified canvases is defined as $\{T_{\|n}\} = \boldsymbol{P}_{\|n} = \boldsymbol{P}^n$, the distribution $\mathcal{P}_{\|n}$ samples n times independently from $\mathcal{P}$;*
- *For each trappified canvas T' defined above and an output $t = (t_j)_{j\in n}$, we have:*
$$\tau'(t) = 0 \textit{ if } \sum_{j=1}^{n} \tau_j(t_j) < w, \textit{ and } 1 \textit{ otherwise} \tag{84}$$
- *The partial ordering of vertices of G^n in $\boldsymbol{P}_{\|n}$ is given by the ordering $\preceq_G$ on every copy of G.*
- *Let $\mathsf{C} \in \mathfrak{C}$. Given a trappified canvas $T_{\|n} = \{T_j\}_{j\in[n]}$, the embedding algorithm $E_{\mathfrak{C},\|\mathfrak{n}}$ applies $E_{\mathfrak{C}}$ to embed C in each T_j.*

The next lemma relates the parameters above to the detection and insensitivity of the compiled scheme.

Lemma 12 (Exponential Detection and Insensitivity from Parallel Repetitions). *Let $\boldsymbol{P}$ be a trappified scheme on graph G which ϵ-detects the error set $\mathcal{E}_1$, is δ-insensitive to $\mathcal{E}_2$ and perfectly insensitive to $\{\mathsf{I}\}$. For $n \in \mathbb{N}$ and $w \in [n-1]$, let $\boldsymbol{P}_{\|n}$ be the trappified scheme resulting from the compilation defined in Definition 20.*

Let $\mathcal{E}_{\geq k,\mathcal{E}}$ and $\mathcal{E}_{\leq k,\mathcal{E}}$ be defined as in Theorem 9. Let $k_1 > w/\epsilon$ and $k_2 < w/\delta$. Then, $\boldsymbol{P}_{\|n}$ $\epsilon_{\|n}$-detects $\mathcal{E}_{\geq k_1,\mathcal{E}_1}$ and is $\delta_{\|n}$-insensitive to $\mathcal{E}_{\leq k_2,\mathcal{E}_2}$ where:

$$\epsilon_{\|n} = \exp\left(-2\frac{(k_1\epsilon - w)^2}{k_1}\right), \tag{85}$$
$$\delta_{\|n} = \exp\left(-2\frac{(k_2\delta - w)^2}{k_2}\right). \tag{86}$$

Proof. As in the proof of the previous lemma, we denote Y a random variable counting the number of trappified canvases whose decision function rejects.

Let $\mathsf{E} \in \mathcal{E}_{\geq k_1,\mathcal{E}_1}$. We can lower-bound Y in the usual stochastic order by a (k_1, ϵ)-binomially distributed random variable $\tilde{Y}$. Then, Hoefffding's inequality yields directly that:

$$\Pr[Y < w] \leq \Pr\left[\tilde{Y} < w\right] \leq \exp\left(-2\frac{(k_1\epsilon - w)^2}{k_1}\right). \tag{87}$$

Similarly, let $\mathsf{E} \in \mathcal{E}_{\leq k_2, \mathcal{E}_2}$. Due to the perfect insensitivity of $\boldsymbol{P}$ to I, we can now upper-bound Y in the usual stochastic order by a (k_2, δ)-binomially distributed random variable $\tilde{Y}$. Then, Hoefffding's inequality yields directly that:

$$\Pr[Y \geq w] \leq \Pr\Big[\tilde{Y} \geq w\Big] \leq \exp\left(-2\frac{(k_1\epsilon - w)^2}{k_1}\right). \tag{88}$$

■

Test and Computation Separation from Parallel Repetitions. We can now recover the case where some runs contain tests only while others consist only of the client's computation. This will be based on the following remark

Remark 6 (Pure Computation). A trappified scheme $\boldsymbol{P}$ may also only contain a single trappified canvas on graph $G = (V, E)$ such that $V_T = \emptyset$. This is the opposite case from Remark 3 above in the sense that all vertices serve to embed a computation of interest and none are devoted to detecting deviations. The decision function always accepts. Correctness is trivially (perfectly) satisfied for set $\{\mathsf{I}\}$, the detection and insensitivity are $\epsilon = 1$ and $\delta = 0$ respectively for any set.

We then use Remarks 3 and 6, which allow us to define trappified schemes $\boldsymbol{P}_C$ and $\boldsymbol{P}_T$ on a graph G. $\boldsymbol{P}_C$ contains a single empty trappified canvas (with no trap) which can then be used to embed any computation on graph G, with 1-detection and 0-insensitivity to all Paulis. On the other hand, $\boldsymbol{P}_C$ may only contain pure traps with no space for embedding any computation, which ϵ-detects a set of errors $\mathcal{E}_1$ and is δ-insensitive to $\mathcal{E}_2$ (and perfectly insensitive to $\{\mathsf{I}\}$).

Then, Lemma 2 allows us to compose these two schemes via a probabilistic mixture noted $\boldsymbol{P}_M$. For parameters $d, s \in \mathbb{N}$ and $n = d + s$, $\boldsymbol{P}_M$ chooses schemes $\boldsymbol{P}_C$ and $\boldsymbol{P}_T$ with probabilities d/n and s/n respectively. The parameters for $\boldsymbol{P}_M$ are $\epsilon_M = (d + s\epsilon)/n = 1 - (1 - \epsilon)s/n$ and $\delta_M = s\delta/n$. It is also perfectly insensitive to $\{\mathsf{I}\}$.

Now the parallel repetition of Lemma 12 can be applied to $\boldsymbol{P}_M$ with parameters n, w to yield $\boldsymbol{P}_{\parallel n}$ with the following parameters:

$$\epsilon_{\parallel n} = \exp\left(-2\frac{(k_1(1 - (1 - \epsilon)s/n) - w)^2}{k_1}\right), \tag{89}$$

$$\delta_{\parallel n} = \exp\left(-2\frac{(k_2 s\delta/n - w)^2}{k_2}\right), \tag{90}$$

for values $k_1 > wn/(n - s + s\epsilon)$ and $k_2 < wn/s\delta$.

Notice that the bound on k_2 is identical to the one from Theorem 9, while the value for k_1 is smaller. The bounds obtained here are also simpler since they do not require an optimisation over the parameter χ, while still being exponential. However, they may be less sharp due to the degradation of ϵ_M (since we consider here the computation as trappified canvases which always accept).

Finally, note that $\boldsymbol{P}_M$ is not strictly speaking a trappified scheme since it cannot embed computations with probability 1 as is required from Definition 8. However, all claims here are valid as they only consider the detection and insensitivity parameters, showing again the importance of separating these three properties. Consequently, the analysis in terms of correctness may slightly more complex since the number of computation runs is probabilistic, but can be bounded using tail bounds.

G A Linear Programming Problem for Trap Optimisation

In particular situations, it might be useful to have more granular control of the design and error-detecting capabilities of the test rounds.

For instance, because of hardware constraints or ease of implementation, it might be favourable to restrict the set of tests one is willing to perform to only a subset of the tests resulting from generalised traps. As one example, one might desire to avoid the preparation of dummy states and therefore restrict the set of feasible tests to those requiring the preparation of quantum states in the $X-Y$-plane only. It might also not be necessary for the employed tests to detect all possible Pauli errors because of inherent robustness of the target computation.

In such cases, we can expect better error-detection rates if we (i) allow for more types of tests, or (ii) remove deviations from the set of errors that are required to be detected. To this end, we present a linear programming formulation of the search for more efficient tests in Problem 1.

Problem 1 Optimisation of the Distribution of Tests

Given
- a set of errors $\mathcal{E}$ to be detected,
- a set of feasible tests $\mathcal{H}$,
- a relation between tests and errors describing whether a test detects an error, $R : \mathcal{H} \times \mathcal{E} \to \{0, 1\}$,

find an optimal distribution $p : \mathcal{H} \to [0, 1]$ **maximising** the detection rate $\epsilon \in [0, 1]$ **subject to** the following conditions:
- p describes a probability distribution, i.e. $\sum_{H\in\mathcal{H}} p(H) \leq 1$,
- all concerned errors are detected at least with the target detection rate, i.e.

$$\forall E \in \mathcal{E} : \quad \sum_{\substack{H\in\mathcal{H} \\ R(H,E)=1}} p(H) \geq \epsilon. \tag{91}$$

Remark 7. While efficient algorithms exist to find solutions to such real-valued constrained linear problems, in this case the number of constraints grows linearly with the number of errors that need to be detected, and therefore generally exponentially in the size of the graph.

Remark 8. Solutions to the dual problem of Problem 1 are distributions of deviations applied to the test rounds. An optimal solution to the dual gives therefore an optimal attack, i.e. a distribution of deviations that achieves a minimal detection rate with the tests at hand.