Verification of Quantum Computations: Hardware-Efficient Security Proofs

Harold Ollivier

Spring 2026

To David and Ray.

Contents

Foreword

It is a well-established, if rarely admitted, truth that the two most compelling sections of an Habilitation à Diriger des Recherches (HDR) are—or should be—this Foreword and the subsequent Acknowledgments.

There are obvious reasons for this: the totality of the work presented herein has already been made publicly available with far greater detail in peer-reviewed venues. Consequently, there is nothing new from a technical perspective in the chapters that follow. New content is thus strictly confined to these two preliminary sections, where I am permitted to speak without the shield of clinical, scientific anonymity.

Indeed, this manuscript is synthesized from the following works: [LMKO21, KKL$^+$24, KKL$^+$25, KLMO24, GLMO24, KLMO25]. While this list is chronological, it does not reflect the order in which the papers appear in this manuscript. Our work in [LMKO21] was initially an effort to ensure that experimentalists would be able to implement verification with noisy devices; however, it was only after completing the project that we realized it contained many ingredients that were significantly under-exploited.

Generalizing these elements led to [KKL$^+$24], which contains the bulk of the new conceptual insights for verification and pushes modularity to its logical limit by providing a security proof compiler—one that can derive security solely from error detection schemes. This is expanded in Part II, which also comprises a useful protocol extracted from [KKL$^+$25] for implementing Remote State Preparation without ever deviating from the equatorial plane of the Bloch sphere. This is a crucial feature that serves many purposes, as will become clear as one progresses through the document.

Section Part III is devoted to using these tools following the original motivation that prevailed in [LMKO21]: making it as light as possible for an experimentalist to implement secure computation. In this sense, [KLMO24, GLMO24] justify the title of this manuscript by demonstrating how improved proof techniques provide secure protocols with lighter hardware requirements. Finally, I conclude with prospective protocols in Part IV exploring multi-party computation [KKL$^+$25] and the delegation of fault-tolerant computations [KLMO25]. Both areas represented significant technical challenges, as the existence of protocols achieving security in these scenarios had remained open questions for nearly two decades.

Since the novelty of this exercise lies solely in the presentation and succession of ideas, I have deliberately omitted all formal proofs, referring the interested reader to the original publications. This is a calculated attempt to find the least-damaging trade-off available to French researchers when they are cornered into producing an HDR.

To put it bluntly, this has not been a pleasant exercise. At times, the process of satisfying administrative obligations feels less like a validation of mentoring quality and more like a form of institutionalized hazing. It is a curious paradox: the requirement for an HDR is to safeguard the quality of PhD supervision, yet it achieves the opposite by diverting hundreds of hours away from actual students toward the paraphrasing of existing work.

To be precise, this manuscript represents over 360 hours of labor—roughly 22% of the official French annual working time. Even when smoothed over the two-year period dedicated to its writing, this 11% tax on my productivity is staggering. For the sake of comparison, this is twice the time I spent working directly with one of my PhD student over the same period. Did this investment improve my mentoring capabilities? Certainly not. To offer another metric: this effort represents half the

yearly energy required to manage the Hybrid Quantum Initiative, a 36M€ R&D project.

The cumulative impact—time stolen from students and energy distracted from managing large-scale structuring projects—represents a massive loss in missed opportunities and, ultimately, public funding. While the international competition demands that we move fast, train students, and lead ambitious projects, we remain anchored by the archaic requirement to hold an HDR simply to create a team or mentor students.

I was fortunate enough to have either the support to bend these rules or the sheer stubbornness to ignore them. This allowed me to establish a research team of over 20 people and hire four PIs—tasks for which, strictly speaking, I was not yet habilitated. Had I followed the injunctions to the letter, over 50 publications since 2023 would have been credited to other institutions.

However, the situation eventually became untenable. My newly hired PIs also face the hurdle of their own HDRs; without my own degree, their ability to mentor students is restricted, harming their careers, their collaborations, and their ability to secure funding. This, and only this, is the reason for this manuscript. Otherwise, I would have been quite happy to continue my administrative guerrilla warfare indefinitely.

Why include such a rant? Because the HDR is a lingering artifact of the 1984 reform. When the Thèse d'État was suppressed to align France with international standards, the HDR was seemingly promoted as a way to undo what had just been done: to reintroduce a hierarchy that ensured the Caciques with a Thèse d'État would not be confused with the new PhDs. It ensured that they could maintain their positions without facing unseemly competition from younger, and often sharper, PIs.

Today, the HDR resembles military service: before you endure it, you know it is maladapted; while you endure it, you confirm that opinion; but once finished, there is a lingering, cynical desire to see the young ones go through it as well—simply because one does not want them to escape the suffering one had to endure.

For over forty years, this situation has irritated generations of researchers. It is perhaps time to move on and focus on what actually matters—mentoring students, advancing research, and doing science—and replacing a static diplomas with real emphasis in the recruitement process on HR qualities required to manage a team.

Acknowledgements

If two figures in my career must stand out, they are those of two remarkable scientists: Pascale Charpin and Elham Kashefi.

Pascale welcomed me into the CODES team at INRIA, displaying a level of trust that, in hindsight, bordered on the heroic. She agreed to supervise a PhD student she did not know, on a subject—the information-theoretical understanding of decoherence—that was entirely foreign to her own expertise in symmetric cryptography. For the subsequent three years, we did not work together at all.

I can only imagine the secondary stress of hosting a student who spent his time abroad, working with researchers outside her community, and occasionally picking up fights with the entire administration and the president of his own thesis jury. While I sincerely hope I never have to supervise a student quite like myself, I must admit it was a dream PhD. Thank you, Pascale, for the rarest of academic gifts: total autonomy.

Elham Kashefi was instrumental in dragging me back to the world of research. After thirteen years spent founding, funding, and exiting companies, I had reached the realization that the corporate world was not where I belonged. The intellectual challenge had lost its luster.

The breaking point arrived in 2014, when I found myself forced to write into a legal contract that -500 is indeed smaller than -300. This was necessary to ensure that neither lawyers nor judges could misinterpret a clause stating: *"In case the EBITDA is below -300 kEur, we shall have the right to convert our bonds."* I though, it was time to do something else.

I never believed a return into reasearch would be possible. Yetm here we are. This return was sparked by a chance encounter, orchestrated by Stéphane Buttigieg—my second recruit at the Institut Louis Bachelier—who recognized my name on a slide Elham had presented at a "Women in Science" conference. Elham took the significant risk of onboarding me into her team, granting me the necessary time to slowly catch up with thirteen years of scientific progress that had passed me by. Once again I was granted trust, autonomy and a wonderful research environment.

I am also deeply indebted to Jean-Frédéric Gerbeau and Bruno Sportisse, who invited me to rejoin INRIA and afforded me the opportunity to build the QAT team. I have been exceptionally lucky to hire incredibly talented colleagues; working with them is a constant pleasure that almost makes the administrative overhead of the French system bearable.

Finally, I want to thank Laure and Sidoine for their continuous support and infinite patience. They know all too well the manic cycles of research: the euphoria of a new proof, the despair of breaking said proof, the frantic excitement of a new new proof, and the disparair....

Cyclicity, it seems, is the defining characteristic of my career.

Part I
Introduction & Challenges

1 Introduction

Modern science rests on a simple loop: predict a phenomenon, perform the experiment, compare numbers. In quantum physics that loop breaks as soon as the device under study becomes computationally rich. Simulating a 50-qubit circuit, or the many-body dynamics of a 200-site spin chain, exhausts classical super-computers; predicting the result of a generic 100 qubit quantum circuit with depth 100 is entirely out of reach. Yet such high-complexity regimes are exactly where the quantum computer promises its greatest impact. How, then, can a classically limited scientist verify claims about a machine that surpasses classical limits?

The tension is already visible in today's quantum advantage experiments. Random circuit sampling on a few dozen qubits can be cross-checked—at the cost of consuming every GPU cluster within reach. Add twenty extra qubits and the classical cross-check evaporates. By contrast, Shor's factoring algorithm permits crisp verification: if the device returns the prime factors of a 2048-bit RSA key, multiplication on a laptop confirms the answer. The difference lies not in the physics of the device but in the complexity class of the problem. Factoring belongs to NP: a classical certificate—the prime factors—can be checked quickly, even if finding it is hard. However, a generic quantum computation—those in the Bounded Error Quantum Polynomial (BQP) time class but not in NP—offer no such classical certificate.

Aharonov and Vazirani [AV13] proposed a way out. They lifted the predict-verify loop into an interactive dialogue between two agents:

- A *verifier*, a Bounded Error Polynomial time (BPP) machine, who may additionally prepare or measure single-qubits but cannot simulate the entire computation.
- A *prover*, an untrusted BQP machine, holding the large quantum computer.

Through a sequence of tailored challenges the verifier convinces himself that the prover executed the desired algorithm and refrained from deviations harming the validity of the reported result. An interactive protocol thus replaces the impossible task of predicting the output of a quantum computation by the feasible task of testing the prover's responses. Two questions follow naturally:

1. Which quantum computations admit an interactive proof? We seek protocols that cover the whole of BQP, not merely special cases like Shor's factoring algorithm.
2. Can these protocols survive the practical hurdles of possibly noisy quantum hardware used by the verifier and/or the prover?

The first question—is verification even possible in principle?—is now essentially settled. The protocols presented in [FK17, ABOE10] prove statistical soundness. Several works have improved upon these initial results to cover a wider range of settings. For instance, quantum capabilities on the verifier's side can be removed with the help of two non-communicating provers [RUV13, GKW15, MF16, NV17], or by resorting to computational hardness assumptions [Mah18]; the verifier can be restricted to performing measurements only [FHcvM18, HKSE17]; while reference [Bro18] provides a circuit-model-based verification protocol where quantum communication is proportional to the number of T gates .

The second question—are these protocols practical enough to be of any use?—is more than just rhetorical. Verification is essential for the development of quantum computing as users accessing a remote machine need a guarantee of integrity for their computation, service providers want to know that the machine they buy is really capable of solving classically intractable problems, and even quantum hardware providers need to test whether their design improvements are going in the

right direction. Yet, protocols cited above fail in two aspects. They lack robustness and/or they impose high overheads on verifiers and provers. In effect, the lack of robustness amounts to mistake honest noise for a malicious behavior forcing aborts, while the overhead imposes to dedicate precious qubits to security at the expense of those that are available for computing.

Closing this robustness and overhead gap is the challenge the rest of this document sets out to address.

Organization and Contributions

In Chapter 2 we review techniques and protocols that provide the necessary background for statistically secure prepare-and-send verification of quantum computations using the approach developed in [FK17]. These provide the necessary ingredients for the extensions presented in this document. In Chapter 3, we expose the challenges faced when implementing verification protocols in future machines and hint at the theoretical developments needed to solve them.

In Part II, we develop our toolbox for addressing the challenges of practical verification. We start by introducing a modular and composable framework that extracts the necessary functionalities to construct verification protocols (Chapter 4). We follow it with two direct applications. The first one (Chapter 5) is the simplest of the newly introduced protocols but it already pushes the boundaries of what can be done in practice. It removes the space overhead for running a verified computation, therefore suppressing the security vs. computation trade-off mentioned earlier. It also allows the prover to be noisy provided the noise stays below a given threshold expressed at the circuit level—and unfortunately not at the gate level. Yet, it shows that some robustness is indeed possible. In the second one (Chapter 6), a seemingly innocuous improvement is performed—reducing the number of states the verifier needs to prepare—from 10 to 8. More than quantitative, the improvement is qualitative as the 8 states are in the XY plane of the Bloch sphere. This property alone, will in fact be used consistently in all the subsequent protocols that we introduce.

In Part III we focus on reducing the physical requirements on the verifier's side. In Chapter 7, we show how to verify quantum computations using rotations and bit flips only—i.e. without the need to prepare quantum states—under the assumption that a possibly maliciously controlled source is sending qubits and not higher dimensional systems. In Chapter 8 we take a complementary approach where the verifier is allowed to access an off-the-shelf attenuated laser.

In Part IV we propose to extend the realm of verification in two regimes. The first one is the multi-party case, where it was known how to lift a classical secure multiparty computation to the quantum domain in a mutli-round symmetrically powerful setup. We show (Chapter 9) that an asymmetric setup with a single powerful server is possible, thereby also simplifying the quantum network topology from being complete to being star-shaped. We then address in Chapter 10 the important question of long-term scalability of these protocols under noise. It was unknown if statistically secure verification protocols could be implemented for error protected quantum computations. The difficulty is that logically encoded quantum computations greatly enlarge the attack surface for the prover to bypass the requirement for a sound interaction with the verifier. More precisely, error correction is creating side-channels that jeopardize previously known protocol designs and proof techniques. Carefully tweaking fault-tolerant compilation procedures led to closing these side channels and allowed to provide the first statistically secure verification protocol for fault-tolerant quantum computations.

In Part V, we provide perspective for future work.

2 Essentials

This chapter establishes the notation, summarizes measurement-based quantum computation (MBQC), and reviews the two foundational secure delegation schemes that underpin the remainder of this work: Universal Blind Quantum Computation (UBQC) and Verified Blind Quantum Computation (VBQC).

2.1 Notation

- *Sets.*
  - 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 set A, $|A|$ denotes the number of elements in A.
  - We denote by Θ the set of angles $\left\{\frac{k\pi}{4}\right\}_{k\in\{0,\dots,7\}}$.
- *Probabilities.*
  - For a distribution probability $\mathcal{D}$, $x \xleftarrow{\$} \mathcal{D}$ indicates that x is sampled from $\mathcal{D}$. Similarly, for a set Λ, $\lambda \xleftarrow{\$} \Lambda$ indicates that λ is sampled uniformly at random from Λ.
  - 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 $\mu(\eta) = 1 - \epsilon(\eta)$.
- *Unitaries and Operators.*
  - We note $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ the Pauli operators. Then $\mathcal{P}_1 = \langle \mathsf{X}, \mathsf{Y}, \mathsf{Z} \rangle$ is the single-qubit Pauli group.
  - The rotation operator around the Z-axis of the Bloch sphere by an angle θ is noted $\mathsf{Z}(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$.
  - The 2-qubit controlled-phase gate is denoted CZ and flips the phase of the $|1\rangle \otimes |1\rangle$ computational basis vector, leaving all others unchanged.
  - Given a set of qubits indexed by elements in set V, for all $i \in V$ and any single-qubit unitary U, we denote U_i the unitary obtained by applying U to qubit i and identity to the rest of the qubits in V. This is easily extended to multi-qubit gates by using multiple indices.
  - Using the previous notation the n-qubit Pauli group is

$$\mathcal{P}_n = \{\pm 1, \pm i\} \times \{\mathsf{P}_1 \otimes \ldots \otimes \mathsf{P}_n \mid \mathsf{P}_j \in \{\mathsf{I}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}\}\}.$$

  - When there is no ambiguity, we will overload the notation P_i to refer to the i-th factor in P whenever P can be written as a tensor product of single-qubit operators, e.g. for $\mathsf{P} = \mathsf{A} \otimes \mathsf{B} \otimes \mathsf{C}$, $\mathsf{P}_2 = B$.
  - The states in the XY plane of the Bloch sphere are noted $|+_\theta\rangle = \mathsf{Z}(\theta)|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$.
  - For a measurement in the basis $|\pm_\theta\rangle$, we associated the value 0 to outcome $|+_\theta\rangle$ and 1 to outcome $|-_\theta\rangle$.
  - For an n-qubit operator U and n-qubit mixed state ρ, we write $\mathsf{U}[\rho]$ for $\mathsf{U}\rho\mathsf{U}^\dagger$.

– For two operators U and V acting on the same number of qubits, we write $U \circ V$ for the composition of the two operators.

Graph States.

- Given a graph $G = (V, E)$ where V is the set of vertices and E the set of edges, $|G\rangle$ is the normalized eigenstate of $X_i \bigotimes_{i\sim j} Z_j$ for $i \in V$ and where $i \sim j$ whenever $(i, j) \in E$.
- We denote the set of neighbors of a node i with $N_G(i) = \{j \in V,\ i \sim j\}$.
- $|G\rangle$ is also defined as $\prod_{(i,j)\in E} CZ_{i,j} \left[\bigotimes_{k\in V} |+\rangle\langle+|_k\right]$ where $CZ_{i,j}$ is the controlled phase gate applied according to the edge (i, j) of G.
- An open graph state with input set $I \subset V$ is obtained by applying the same CZ gates as for regular graph states, the only difference is that a state $|\psi\rangle$ over the qubits in I is provided instead of $\bigotimes_{k\in I} |+\rangle_k$ (Fig. 2.1).

Complexity

- BQP, for Bounded-Error Quantum Polynomial-Time, is the class of decision problems solvable by a uniform family of polynomial-size quantum circuits, with at most $1/3$ probability of error.
- BPP, for Bounded-Error Probabilistic Polynomial-Time, is the class of decision problems solvable by a non-deterministic polynomial time machine such that: if the answer is 'yes' then at least $2/3$ of the computation paths accept; if the answer is 'no' then at most $1/3$ of the computation paths accept.

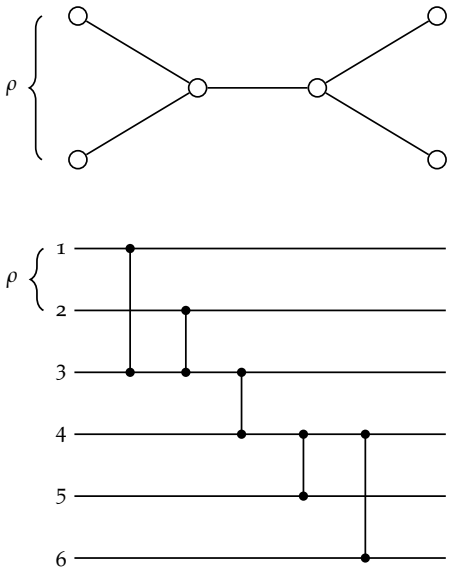


Figure 2.1: Open graph state (up) and corresponding circuit model preparation (down).

2.2 Measurement Based Quantum Computation (MBQC)

MBQC is an alternative model for quantum computing. To perform a computation in MBQC, we need to prepare large entangled state, measure selected qubits, and apply feed-forward Pauli corrections fixed by previous outcomes.

This model emerged from the gate teleportation principle (Fig. 2.2). It was introduced in [RB01]. The measurement calculus [DKP07] formalized the procedure and the correspondence with the circuit model.

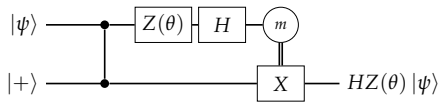


Figure 2.2: Gate Teleportation

Computations as Patterns

The core of MBQC is to perform a computation by following these three steps:

1. Construct a graph state $|G\rangle$ associated to the graph $G = (V, E)$.
2. Successively perform single-qubit measurements on a subset of this state;.
3. Update the subsequent measurements after each new single-qubit measurement is performed.

A computation in the MBQC model is defined by a *measurement pattern*:

Definition 1. Measurement Pattern

A *measurement pattern* is a tuple $(G, I, O, \{\phi(i)\}_{i\in O^c}, f)$ where:

- $G = (V, E)$ is a graph.
- $I \subset V$ and $O \subset V$ are input and output vertex sets.
- $\{\phi(i)\}_{i\in O^c}$ assigns an angle to non-output vertices.
- $f : O^c \mapsto I^c$ is an injective *flow* function inducing a partial order $\prec$ on the vertices V.

The computation is executed by measuring each qubit successively using the partial order defined by f. For qubit i, the measurement basis is $\left\{\left|+_{\phi'}\right\rangle, \left|-_{\phi'}\right\rangle\right\}$. The outcome of the measurement at vertex i is denoted $b(i)$ and, keeping our convention, $b(i) = 0$ is associated to $\left|+_{\phi'}\right\rangle$ and $b(i) = 1$ to $\left|-_{\phi'}\right\rangle$. The measurement angle $\phi'(i)$ is dependent on $\phi(i)$ and also on the outcomes $b(l)$ of previous measurements. More precisely, to each vertex i are associated the sets $S_X(i) = f^{-1}(i)$ and $S_Z(i) = \{j, i \in N_G(f(j)),\}$ which are respectively called the X and Z dependency sets for vertex i. A measurement outcome of 1 in a qubit from $S_X(i)$ will multiply the angle of i by -1, while the Z dependencies add π to the angle. Hence

the measurement angle for qubit i is

$$\phi'(i) = (-1)^{s_X(i)}\phi(i) + \pi s_Z(i), \text{ where} \tag{2.1}$$

$$s_X(i) = \bigoplus_{j \in S_X} b(j), \tag{2.2}$$

$$s_Z(i) = \bigoplus_{j \in S_Z} b(j). \tag{2.3}$$

The existence of a flow function f guarantees that patterns can be executed so that the output is independent of the intermediate measurement outcomes. The conditions on f are that it is an injective function from non-output vertices O^c to non-input vertices I^c. The reason for this choice of domain and co-domain is that outputs are not measured and therefore do not generate corrections, while inputs are measured first and therefore do generate corrections. Further details regarding the definition of the flows and its generalizations can be found in [DK06, BKMP07, Sim21].

Graph Transformations as Patterns

Using MBQC, it is also possible to describe some manipulations on graph states or open graph states—i.e. graph states with input vertices where qubits are provided externally instead of being prepared in $|+\rangle$.

- *Breaking.* The *break operator* removes a non-input vertex of G and all incoming edges to this vertex. It works by preparing the vertex in $|0\rangle$. (Fig. 2.3)
- *Bridging.* The *bridge operation* acts on a non-input vertex of degree two. By measuring such vertex in the Y basis and by applying appropriate corrections on its neighbors, one can produce the open graph state corresponding to G' obtained from G by removing the measured vertex and its incoming edges, and instead linking its neighbors directly (Fig. 2.4).

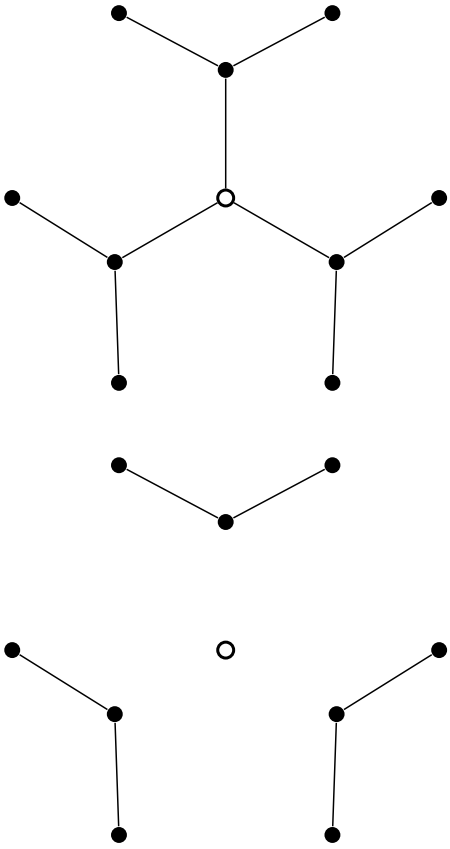


Figure 2.3: Breaking: The initial graph (up) is broken by preparing the empty node in $|0\rangle$ state (down).

Computing on the 3-Qubit Line Graph

To make this more concrete, we will now describe an example of an MBQC pattern and corrections on the three-vertex linear graph. In that case we have $V = \{1,2,3\}$ and $E = \{(1,2),(2,3)\}$. The first qubit in the line will be the only one in the input set I and the last qubit the only one in the output set O. We note $\phi(1)$ and $\phi(2)$ the measurement angles of the first two (non-output) qubits. We start with a single-qubit in state $|\psi\rangle$ as input, the qubits associated to the other two vertices are initialized in the $|+\rangle$ state. We apply one CZ gate for each pair of qubits whose associated vertices are linked by an edge in E. If $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the resulting state is

$$\begin{aligned} \mathsf{CZ}_{1,2}\mathsf{CZ}_{2,3}|\psi\rangle|+\rangle|+\rangle = &\frac{\alpha}{2}(|000\rangle + |001\rangle + |010\rangle - |011\rangle) \\ &+ \frac{\beta}{2}(|100\rangle + |101\rangle - |110\rangle + |111\rangle). \end{aligned} \tag{2.4}$$

In order to perform the measurement on the qubit in vertex 1, we apply the rotation $\mathsf{Z}(-\phi(1))$ and project either onto state $|+\rangle$, associated to the measurement outcome 0, or $|-\rangle$ associated to outcome 1. The joint state of the unmeasured qubits—vertices 2 and 3—is then

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(\alpha + \beta e^{-i\theta})|0\rangle|+\rangle + \frac{1}{\sqrt{2}}(\alpha - \beta e^{-i\theta})|1\rangle|-\rangle, \tag{2.5}$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(\alpha - \beta e^{-i\theta})|0\rangle|+\rangle + \frac{1}{\sqrt{2}}(\alpha + \beta e^{-i\theta})|1\rangle|-\rangle. \tag{2.6}$$

There are two things that we can notice from this:

- In the first case, the state is the same as if we had started with the state $|\psi'\rangle = \mathsf{HZ}(-\phi(1))|\psi\rangle$ and entangled it to a single $|+\rangle$ state using a single CZ operation—a two-qubit linear graph. This fact allows us to perform the translation between the MBQC model and the circuit model.

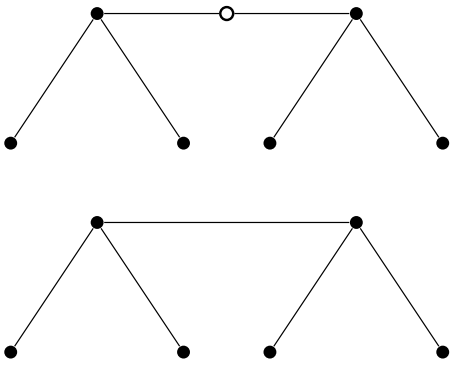


Figure 2.4: Bridging: The initial graph (up) is bridged by measuring the empty node in Y (down).

- We see also that, in order to recover $|\psi_0\rangle$ from the state $|\psi_1\rangle$, we need to apply an X operation on the qubit associated to vertex 2 and a Z operation on the qubit associated to vertex 3. After applying these operations, the state will be independent of the outcome of the measurement. This induces an ordering on the vertices since 1 must be measured before 2 and 3 if we want to use this correction strategy.

These corrections can indeed be absorbed into the angle of future measurements since $\langle\pm|\,\mathsf{Z}(\phi)\mathsf{X} = \langle\pm|\,\mathsf{Z}(-\phi)$ and $\langle\pm|\,\mathsf{Z}(\phi)\mathsf{Z} = \langle\pm|\,\mathsf{Z}(\phi+\pi)$. This can also be done for the qubit in vertex 2. For the output qubits, the corrections need to be applied and cannot be absorbed into a subsequent measurement as, by construction, there are no measurement on output qubits.

In this example above, the flow function is $f(1) = 2$ and $f(2) = 3$, which induces the measurement order $1 \preceq 2 \preceq 3$. More generally, finding the flow relies on the stabilizers of the graph state associated to G [BKMP07, MPS22].

2.3 Delegation of Quantum Computations

A measurement-based quantum computation can be executed remotely. The heavy lifting—preparing a large entangled graph state and performing single-qubit measurements—rests with the server. The client needs only to:

1. Send the classical description of the pattern.
2. Provide its input qubits, if any.

This is summarized in the following protocol, where Alice plays the client and Bob the server:

Protocol 1. Delegated MBQC Protocol

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

Open Graph State Preparation:

1. Alice sends the graph's description (G, I, O) to Bob.
2. Alice sends its input qubits for positions I to Bob.
3. Bob prepares $|+\rangle$ states for qubits $i \in I^c$.
4. Bob applies a CZ gate between qubits i and j if (i, j) is an edge of G.

Computation:

1. Alice sends the measurement angles $\{\phi(i)\}_{i\in O^c}$ along with the description of f to Bob.
2. Bob measures the qubits $i \in O^c$ in the order $\preceq$ induced by f in the basis $\left|\pm_{\phi'(i)}\right\rangle$ where

$$s_{\mathsf{X}}(i) = \bigoplus_{j\in S_{\mathsf{X}}(i)} b(j),\ s_{\mathsf{Z}}(i) = \bigoplus_{j\in S_{\mathsf{Z}}(i)} b(j), \tag{2.7}$$

$$\phi'(i) = (-1)^{s_{\mathsf{X}}(i)}\phi(i) + s_{\mathsf{Z}}(i)\pi, \tag{2.8}$$

   where $b(j) \in \{0, 1\}$ is the measurement outcome for qubit j.
3. Bob applies the correction $\mathsf{Z}_i^{s_{\mathsf{Z}}(i)}\mathsf{X}_i^{s_{\mathsf{X}}(i)}$ for each output qubits $i \in O$, which it sends back to Alice.

2.4 Blind Delegation

Privacy is the first line of defense in delegated quantum computing. *Blindness* provides privacy by ensuring that the server learns nothing about either the client's data or the algorithm beyond a small, explicitly declared leakage.[1] Correctness is not enforced at this stage; an honest-but-curious server returns the right answer, but a malicious one might return garbage without detection.

We formalize blindness as an ideal cryptographic resource, then present the Universal Blind Quantum Computation (UBQC) protocol that constructs the resource, and then illustrate it on a three-qubit example. This formalization follows the principles of Abstract Cryptography [MR11] where security stems from the inability for

[1] It can be set to the size of the graph and the measurement order, if the delegated computation is compiled to rely a universal graph state. This is the least that can leak as the server will always be communicated the graph and the measurement order as it is precisely what it can always deduce from the instructions given by the client.

an all-powerful distinguisher having access to all interfaces involved in the ideal resource and the protocol to distinguish between them (See Chapter A for more details).

Blindness

In the language of abstract cryptography, blindness of computation is captured by the following resource, where Alice plays the role of a client who delegates her computation to Bob, playing the server:

Resource 1. Blind Delegated Quantum Computation (BDQC)

Public Information: Nature of the leakage l.
Permitted leakage l: A set of computations $\mathfrak{C}$, and two subspaces, one for inputs $\Pi_{I,\mathfrak{C}}$ and one for outputs $\Pi_{O,\mathfrak{C}}$, so that for $\mathsf{C} \in \mathfrak{C}$, an input in $\Pi_{I,\mathfrak{C}}$ is mapped to an output in $\Pi_{O,\mathfrak{C}}$.
Inputs:

- Alice inputs the classical description of a computation C from subspace $\Pi_{I,\mathfrak{C}}$ to subspace $\Pi_{O,\mathfrak{C}}$ and a quantum state ρ_A in $\Pi_{I,\mathfrak{C}}$.
- Bob chooses whether or not to deviate. This interface is filtered by a control bit c (set to 0 by default for honest behaviour). If $c = 1$, Bob has an additional input CPTP map F and state ρ_B possibly entangled with its own working register.

Computation by the Resource:

1. If $c = 1$, the Resource sends the leakage l to Bob's interface.
2. If $c = 0$, it outputs $\mathsf{C}[\rho_A]$ at Alice's output interface. Otherwise, it waits for the additional input and outputs $\mathsf{F}[\rho_{AB}]$ at Alice's interface.[2]

Above the leakage is the graph state and order of measurement that would be used to perform the corresponding MBQC. Note that if one uses a universal graph, then the leakage reduces to an upper bound on the size of the computation. The subspaces $\Pi_{I,\mathfrak{C}}$ and $\Pi_{O,\mathfrak{C}}$ are only specifying how the input and output are encoded into qubit systems.

A direct inspection shows that this definition follows the textual description of what a blind delegated quantum computation is supposed to do:

- Alice delegates the execution of the computation C on ρ_A.
- In case $c = 0$, Bob is honest and Alice receives $\mathsf{C}[\rho_A]$.
- In case $c = 1$, Bob is malicious, Alice receives an arbitrary deviated state, but Bob learns at most l.

Because the resource's behavior strictly abides by its specification, it is said to be *secure-by-design*.

Protocol for Blind Delegated Quantum Computation

Blind Delegated Quantum Computation (Resource 1) can be constructed from single-qubit preparations, $\mathsf{Z}(\theta)$ and X unitaries as well as quantum communication. The intuition for the construction is to rely on the client to encrypt its inputs and the ancillary qubits used to perform the computation by applying random $\mathsf{Z}(\theta)$ rotations and X^a flips independently for each of them (Fig. 2.5). This will allow the client to one-time pad its inputs and the measurement angles. Commuting the $\mathsf{Z}(\theta)$ rotations through the CZ gates simply induces corresponding rotations right before the measurements, while commuting the X^a flips induces additional Z^a on the neighbors. Both can be absorbed by adapting the measurement angle accordingly, resulting in seemingly random instructions sent to the server. The measurement outcomes can also be hidden behind a fresh random bit r, and recovered by the client by undoing this last one-time pad when computing the updated angles for the subsequent measurements. All in all, the computation can be carried out correctly whenever the server follows the instructions and without leaking neither inputs, outputs nor measurements angles of the pattern. In essence, only the graph used and the order of measurement is known to the server.

Protocol 2. Universal Blind Quantum Computation (UBQC)

[2] Here, ρ_{AB} is the joint state of registers A and B. It is not possible to write it as $\rho_A \otimes \rho_B$ as both resgiters could be entangled with one another, and with an external register such as the private register of Bob. This is especially relevant for the security proofs as Alice and Bob are played by a single party, the distinguisher.

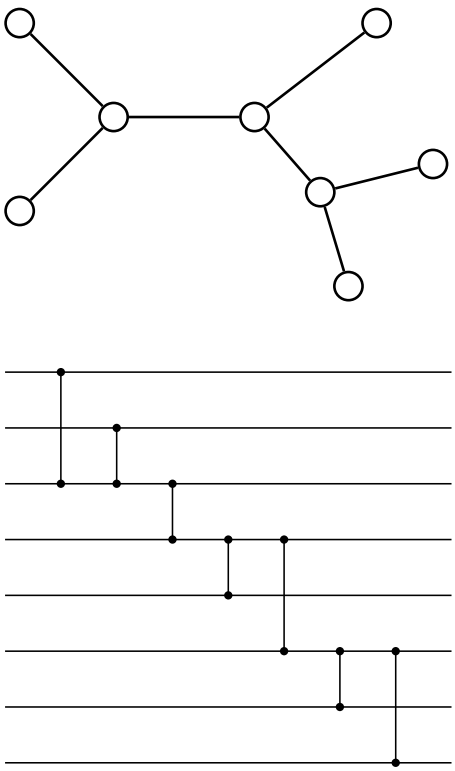


Figure 2.5: Initial graph state (up), circuit preparing the graph state (middle), equivalent circuit with inserted random $\mathsf{Z}(\theta)$ rotations (colored boxes) compensated by the conjugated rotation $\mathsf{Z}(-\theta)$ before the measurement (down).

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

1. Alice sends the graph's description (G, I, O) and the measurement order to Bob.
2. Alice prepares and sends all the qubits in V to Bob:
   (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 $\prod_{i\in I} \mathsf{Z}_i(\theta(i))\mathsf{X}_i^{a(i)}[\rho_A]$. For $i \in I^c$ it sends $\left|+_{\theta(i)}\right\rangle$.
3. The Bob 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, Alice computes the measurement angle $\delta(i)$ and sends it to Bob, receiving in return the corresponding measurement outcome $b(i)$:

$$s_{\mathsf{X}}(i) = \bigoplus_{j\in S_{\mathsf{X}}(i)} b(j) \oplus r(j),\ s_{\mathsf{Z}}(i) = \bigoplus_{j\in S_{\mathsf{Z}}(i)} b(j) \oplus r(j), \tag{2.9}$$

$$\delta(i) = (-1)^{a(i)}\phi'(i) + \theta(i) + (r(i) + a_N(i))\pi, \tag{2.10}$$

   where $\phi'(i)$ is computed using Equation 2.8 with the new values of $s_{\mathsf{X}}(i)$ and $s_{\mathsf{Z}}(i)$.
5. The Bob sends back the output qubits $i \in O$.
6. Alice applies $\mathsf{Z}_i^{s_{\mathsf{Z}}(i)+r(i)}\mathsf{X}_i^{s_{\mathsf{X}}(i)+a(i)}$ to the received qubits $i \in O$.

As announced above, the uniform randomness of $\theta(i)$ perfectly hides the value of $\phi'(i)$ in $\delta(i)$, while the value $r(i)$ hides the measurement outcome but also the output of the computation since it is propagated by the flow and results in a Quantum One-Time-Pad of the output.

In the original protocol from [BFK09], the outputs are prepared by the server in the $|+\rangle$ state and are encrypted by the computation flow. The additional randomization of the output qubit might seem superfluous since the server can simply measure the qubit to recover the value of the state. However, including this additional randomization makes a smoother exposition of verification protocols as they can be directly built on top of UBQC (Protocol 2).

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 security properties of UBQC can be summarized as follows:

Theorem 1. Security of UBQC

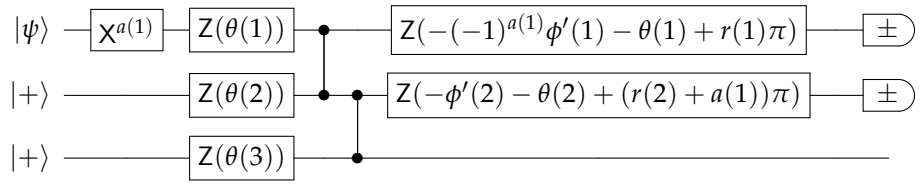
UBQC (Protocol 2) perfectly constructs Resource 1 in the abstract cryptography framework.

Blind Computing on the 3-Qubit Line Graph

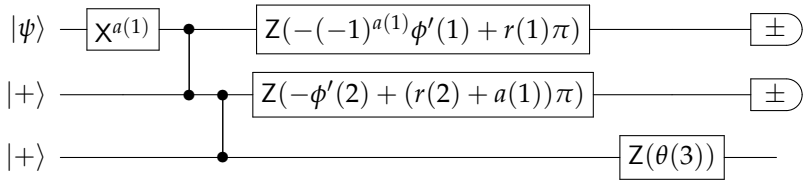
We use the example described above to demonstrate an execution of this protocol. Alice would like to hide her input state $|\psi\rangle$, the measurement angles $\phi(1)$ and $\phi(2)$ and the output. For the input, Alice uses a variant of the Quantum One-Time-Pad, sampling a random bit $a(1) \in \{0, 1\}$ and a random angle $\theta(1)$ and applying the operation $\mathsf{Z}_i(\theta(i))\mathsf{X}_i^{a(i)}$ to $|\psi\rangle$. For the non-input qubits she sets $a(2) = a(3) = 0$. For the non-output qubit she samples at random $\theta(2) \in \Theta$ and for the output vertex she samples at random $\theta(3) \in \{0, \pi\}$. she creates the states $\left|+_{\theta(2)}\right\rangle$ and $\left|+_{\theta(3)}\right\rangle$ and sends these two states and its encrypted input to Bob.

Bob receives these three qubits and performs the entangling operations $\mathsf{CZ}_{1,2}$ and $\mathsf{CZ}_{2,3}$ as above. For now the security is guaranteed since the input is perfectly encrypted and the other qubits are in random states uncorrelated to the computation. However, Alice still desires to run her computation and must do so through the encryption. To do so, she will instruct Bob to measure the qubits 1 and 2 with the angle $\delta(i) = (-1)^{a(i)}\phi'(i) + \theta(i) + (r(i) + a_N(i))\pi$, where $a_N(i)$ is the sum of values of $a(j)$ for j neighbours of i and $r(i)$ is a random bit.

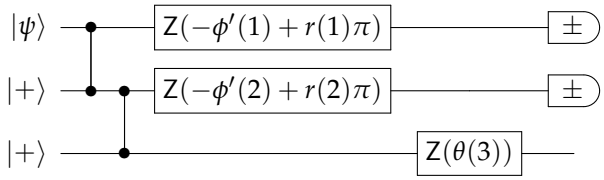
We can see that this performs the same computation as the MBQC example described above. The Z rotation encryption commutes with the CZ operations and cancels out the encryption of the measurement angle. On the other hand, when the X encryption of the input commutes with the CZ gates, it creates an additional Z on the neighbor which is then taken care of by $a_N(2) = a(1)$. Commuting the X to the end also flips the sign of the measurement angle, which is why $(-1)^{a(i)}$ appears in from of $\phi'(i)$ in the expression of $\delta(i)$. In the end the computation is the same as the unencrypted one above so long as Alice corrects the measurement outcomes returned by Bob to account for the additional $r(i)$ by flipping the outcome $b(i)$ if $r(i) = 1$. If $\theta(3) = \pi$, Alice must also apply Z to the output to compensate. This process is summarized in Figure 2.6.



(a)



(b)



(c)

Figure 2.6: Correctness of UBQC for three-vertex linear graph. (a) UBQC Protocol with the explicit values of $\delta(i)$. (b) The Z encryption commutes through the CZ gates and is cancelled out by the later Z rotation. (c) The input X encryption commutes through the CZ gates but adds a Z on qubit 2, which is canceled out by the $a(1)\pi$ inside the rotation. The X on qubit 1 is commuted through the rotation and absorbed by the measurement. The result is Alice's desired MBQC computation up to Pauli corrections and bit-flips.

2.5 *Verified and Blind Delegation*

Integrity complements privacy by allowing the verifier—formerly the client in the blind delegation scenario—to detect deviations in the delegated computation. Verifiability and blindness are the combination of both guarantees: the prover—formerly the sever—learns nothing beyond an explicitly declared leakage and cannot alter the outcome of the delegated computation without being caught. Hence, any malicious behavior either has not effect on the computation or forces the verifier to abort.

We formalize the verifiability and blindness as a resource—the *Secure Delegated Quantum Computation* resource (SDQC)—involving two parties, a verifier and a prover. We then outline the dotted-triple-graph construction of [KW17] that realizes it. Security intuition and formal theorem conclude the subsection.

Verifiability and Blindness

Although verifiability and blindness are distinct and unrelated properties of delegated quantum computation protocols, they often come together. Indeed, as we will

see later, verifiability is obtained with the help of blindness, allowing to interleave the computation of interest with small predictable ones that serve to test the beahvior of the prover. The combination of both properties is captured by the following resource:, where Alice is playing the role of the verifier and Bob that of the prover:

Resource 2. Secure Delegated Quantum Computation (SDQC)

Public Information: Nature of the leakage l.
Permitted leakage l: A set of computations $\mathfrak{C}$, and two subspaces, one for inputs $\Pi_{I,\mathfrak{C}}$ and one for outputs $\Pi_{O,\mathfrak{C}}$.
Inputs:

- Alice inputs the classical description of a computation C from subspace $\Pi_{I,\mathfrak{C}}$ to subspace $\Pi_{O,\mathfrak{C}}$ and a quantum state ρ_A in $\Pi_{I,\mathfrak{C}}$.
- Bob chooses whether or not to deviate. This interface is filtered by two control bits (e, c) (set to 0 by default for honest behavior).

Computation by the Resource:

1. If $e = 1$, the Resource sends the leakage l to Bob's interface; if it receives $c = 1$, the Resource outputs $|\perp\rangle\langle\perp| \otimes |\mathsf{Abort}\rangle\langle\mathsf{Abort}|$ at Alice's output interface.
2. Otherwise it outputs $\mathsf{C}[\rho_A] \otimes |\mathsf{Accept}\rangle\langle\mathsf{Accept}|$ at Alice's output interface.

The leakage l is defined as for blindness and consists of the graph that would be used for the computation and the corresponding measurement order. Similarly, the subspaces $\Pi_{I,\mathfrak{C}}$ and $\Pi_{O,\mathfrak{C}}$ specify how the input and output information is encoded into the graph state.

As we can readily notice, this resource is secure-by-design:

- Bob gets at most the leakage l.
- Whenever it decides to cheat by setting $e = 1$, he can then only force to abort the computation by setting $c = 1$. In case $e = 0$ or $c = 0$, the result of the computation that is transmitted to Alice is correct.

Protocol for Verified Blind Delegated Quantum Computations

We present the dotted-triple-graph VBQC protocol of [KW17]. It is an optimized construction of SDQC (Resource 2) that shares many features with the original protocol described in [FK17]. It has the advantage of yielding streamlined arguments and less complex pictorial representations.

We start by considering that the computation that the verifier wants to perform is protected by an error-correcting code of distance $d_{\min}$ in a fault-tolerant way, and is described as an MBQC pattern running on the graph state defined by $G = (V, E)$. There, the subspaces $\Pi_{\mathfrak{I},\mathfrak{C}}$ and $\Pi_{O,\mathfrak{C}}$ are non-trivial and describe how the verifier's qubits are embedded into the error-correcting code. This being given, we build the *Triple Graph* $T(G) = (T(V), T(E))$. It is obtained from G by replacing every vertex $v \in V$ by a set of three vertices $P_v = (v_1, v_2, v_3)$, called *primary vertices*, and each edge $(v, w) \in E$ by the nine edges (v_i, w_j) for $i, j \in \{1, 2, 3\}$. In effect, this step constructs a graph, with three times the initial number of vertices, and that contains $3^{|E|}$ overlapping copies of G as subgraphs. Then we built the *Dotted-Triple Graph* $DT(G)$ by replacing each edge $e_{i,j} = (v_i, w_j) \in T(E)$ by a vertex v_e and two edges $(v_i, v_{e_{i,j}})$ and $(v_{e_{i,j}}, w_i)$. These are called the *added vertices*. The previous transformations are summarized in Fig. 2.7.

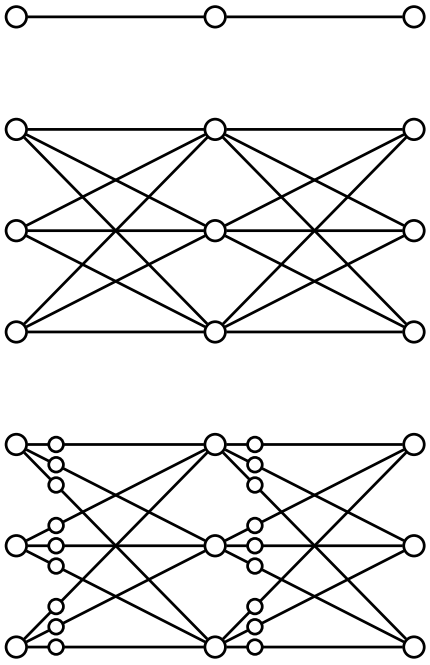


Figure 2.7: The graph G (up), the corresponding triple graph (middle) and dotted triple graph (down).

To delegate the computation in a verifiable fashion, the verifier will assign three possible labels—computation, dummy and trap—for each vertex of the graph. More precisely, for each set P_v, there must be one computation, one dummy and one trap vertex assigned at random. The type of the added vertices is chosen so that the two computation vertices are linked with another computation vertex and two dummies are linked by a trap. All other added vertices are dummies. (See Fig. 2.8).

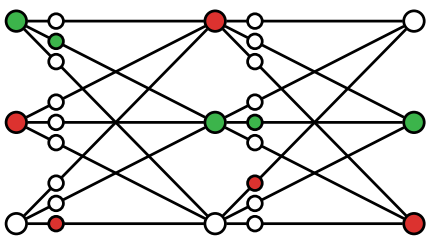


Figure 2.8: A possible assignment of computation (green), trap (red) and dummy (white) vertices on a subgraph of the dotted triple graph corresponding to two consecutive edges in the original graph.

By performing these steps, we have inserted dummies that can be prepared by the verifier in order to implement break operators leaving trap vertices isolated from the rest of the remaining graph. The added vertices with a computation label can then be used to perform bridge operations and recover G, and finish the computation (See Fig. 2.9). The trap vertices are then measured in the basis where they

have been prepared and their measurement checked. In case there is a discrepancy between the outcome and its expected value the computation is aborted.

This corresponds to the following protocol:

Protocol 3. Verified Blind Quantum Computation (VBQC)

Public Information: A graph G and a fault-tolerant error-correcting scheme with distance $d_{\min}$.
Alice's Input: A computation C encoded in a quantum error-correcting code and that can be implemented by an MBQC pattern on a graph G.
The following steps are performed using UBQC
Preparation:

- Input qubits are provided by Alice;
- All the non-input qubits corresponding to computation or trap vertices are prepared in $|+\rangle$;
- All qubits corresponding to dummy vertices are prepared in $|0\rangle$.

Pattern execution:

- Added qubits are measured: computation ones in Y, so as to perform a bridge operation, traps in X and dummies with a random angle chosen from Θ,
- Primary vertices are measured: computation ones according to the pattern taking into account dependencies from the original pattern and those from the bridge measurements, traps in X, dummies with a random angle chosen from Θ.
- For the last layer of primary vertices, Alice performs the measurement of the trap qubits.

Verification: If the values of the trap measurements are all 0 corresponding to obtaining the outcome $|+\rangle$, Alice accepts and keeps the output qubits. Otherwise it aborts.

Note that, compared to UBQC (Protocol 2), the preparation now requires not only $|+_\theta\rangle$ states but also computational basis states $|0\rangle, |1\rangle$.

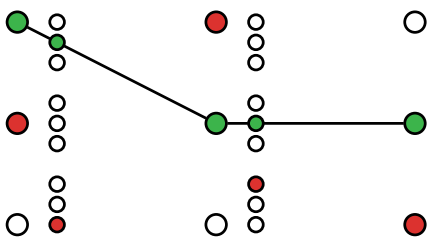


Figure 2.9: Dotted triple graph after breaking operations at the dummy qubits (up) and after the bridging operations are performed (down). Note that the bridging operations are indeed performed blindly while the computation is carried on so that the prover cannot deduce where the computation and trap qubits are located.

Security

In the protocol above, blindness is inherited from UBQC and preserved by the bridge and break operations: these do not reveal locations of computation, traps or dummy qubits.

Verifiability is then a consequence of the predictability of the trap outcomes when decoded by Alice, as well as ignorance of their location for Bob. As a matter of fact,

- each qubit is a trap with probability at least $1/9$ so that the most efficient single-qubit attack has a probability at most $8/9$ of being undetected.
- the fault-tolerant encoding with minimum distance $d_{\min}$ forces Bob to attack at least $c/2$ positions to change the result of the computation. Hence, the probability of failing to detect a *harmful* deviation—a deviation that would change the result of the unencoded computation—scales as $\left(\frac{8}{9}\right)^{c/2}$.

The AC security of Protocol 3 can be shown using the techniques from [DFPR14] and yields:

Theorem 2.

Protocol 3 ϵ-constructs SDQC (Resource 2) for arbitrary quantum computations with ϵ negligibly small in the distance $d_{\min}$ of the error-correcting code, while only leaking the graph $G = (V, E)$ and the order of the measurements used in the MBQC computation.

3 Challenges

The current generation of verification protocols shows that it is possible for a computationally weak verifier to detect if a prover is dishonest. That achievement is intellectually reassuring, at least from a philosophy of science perspective. Yet the associated techniques remain far too cumbersome for deployment on any foreseeable hardware. Overheads in qubit count and extreme sensitivity to noise keep these schemes into the realm of *Gedanken* experiments. Practical roll-out demands sharper theoretical tools.

This chapter isolates the obstacles that matter most—not an exhaustive catalog of open questions, but the bottlenecks that must be cleared before verification can migrate from theory pages to laboratory racks and beyond. It also offers perspectives on doing more with the same techniques and resources, making the approach more attractive for hardware vendors to incorporate into their roadmaps.

3.1 Lack of a modular and composable framework

Early verification schemes were crafted case-by-case. Each paper fixed a particular trust model, chose its own security definition, and proofs were monolithic arguments, sometimes redoing what was done already elsewhere, save for some optimizations. Cross-referencing those proofs—or even lining up their security definitions—quickly becomes an exercise in translation. This ad-hoc landscape has two practical drawbacks:

- *Protocol design stagnates.* When blindness, trap testing, and error correction are intertwined, tuning any single ingredient forces a return to first principles. A higher-dimensional code, a leaner trap schedule, or a different communication pattern all trigger a full re-proof. The result is slow iteration and limited cumulative progress.
- *Composing protocols is generally unsafe.* Quantum accelerators will likely service only a slice of a larger hybrid workflow. If the quantum slice is verified, then its proofs and indeed all its components need to be composable. This not only ensures the integrity of the quantum computation, but also renders the protocols secure even when used inside hybrid workflows. This ensure that modular engineering can be applied at this larger scale.

Without a framework that isolates the various required functionalities, both researchers and practitioners face an explosion of bespoke proofs. Chapter 4 introduces such a framework. It casts each capability as an abstract cryptography resource with clear interfaces, allowing future optimizations to snap in without reopening the entire security ledger. In addition, it provides a compiler which eliminates the need for the protocol designer to go into security proofs but instead replaces it with checking properties of error detection codes. [DFPR14] recognized the need for providing composability but took a slightly different route than the one we present: its objective was to show how non-composable proofs with *local* criteria can nonetheless yield composable security when additional constraints are added.

3.2 Prover side overhead

Graph size inflation

The crux for turning the UBQC protocol into a verification protocol is to insert traps while maintaining both universality and blindness. This heavily relies on the bridge and break operators that allow to start from a larger graph and reduce it to the desired one. Because both operators are single-qubit transforms that are expressible within the XY plane, they respect the blindness guarantees of UBQC.

The canonical recipe from [FK17] begins with a dotted-complete graph able to host three disjoint copies of the target computation graph G. One copy executes the algorithm; the other two carry traps, one on the original vertices and one on the dotted vertices. Traps thus probabilistically blanket every physical location (See Fig. 3.1).

The main problem with the dotted-complete construction is that it can introduce up to $O(|V|^2)$ additional edges, a quadratic swell that might cancel out any effort to use quantum computers for advantages below quadratic. Several refinements have been proposed to mitigate this overhead :

- Hybrid protocols that invoke VBQC only on small sub-instances such as those presented in [KDK15].
- The dotted-triple graph of [KW17], which replaces global completeness with structured triples.

Unfortunately, they don't fully remove the overhead on the prover's side, yielding a trade-off between the security guarantee and the number of qubits available for computing. The circuit-model variant [Bro18] likewise aims to lower the overhead for the prover, but does not provide composability, which hinders its practical usefulness.

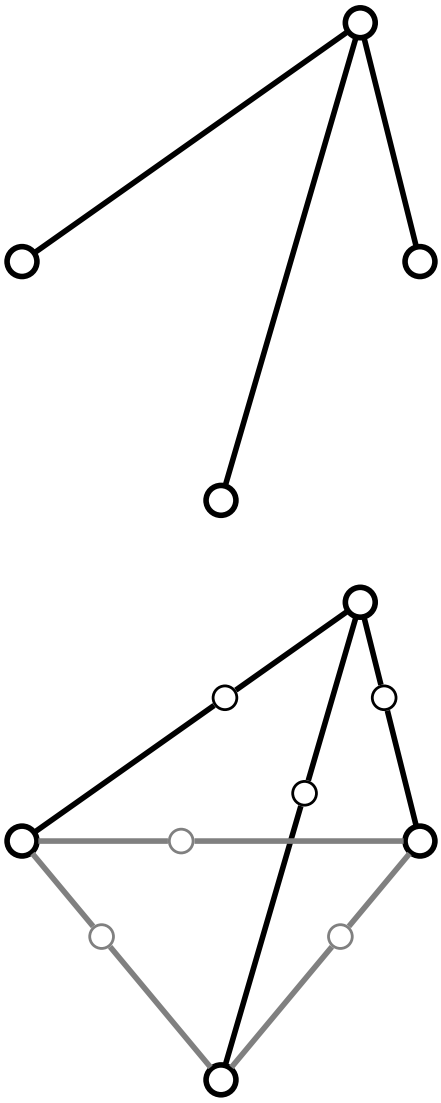


Figure 3.1: Initial graph (up) corresponding dotted-complete graph (down).

Security Amplification Demands Heavy Machinery

To suppress the probability of undetected deviation, VBQC embeds the logical circuit in an error-correcting code and requires three dotted copies of this encoded graph. The logic is straightforward:

- A low-weight attack on encoded data is corrected by the code.
- A high-weight attack must also disturb several traps, whose locations are hidden by blindness and therefore unpredictable.
- Even if each individual trap detects with constant probability, the probability that harmful deviations are undetected falls exponentially with the code distance.

As a consequence, this overhead grows with the desired security parameter: a larger code distance or a lower targeted security error translates directly into more physical qubits and gates on the prover side. For providers with fixed hardware budgets, the consequence is again a smaller effective computing power.

A smoother path would decouple amplification from graph size. The authors of [KW17] note that classical repetition suffices for amplifying purely classical computations, yet it keeps the triple graph unchanged for the classical case. Chapter 5 revisits this choice and shows that repetition alone can deliver cryptographic security with no qubit overhead per repetition when compared to the unprotected computation

3.3 Verifier side overhead

From the theorist's perspective, UBQC and VBQC protocols are lightweight for the verifier. Yet the hardware it presupposes is non-trivial. The verifier must:

- Prepare single-qubits—say photons—on demand.
- Choose one of 10 states $\{|+\rangle_\theta, \theta \in \Theta\} \cup \{|0\rangle, |1\rangle\}$.
- Deliver those states to the prover with sub-gate timing precision and deterministically.

Single-photon sources that approach these specifications exist, but they are bulky, cryogenic, and far from commodity devices. These requirements therefore block even small-scale proof-of-concepts.

A radical alternative is to remove the verifier's quantum duties altogether. Classical-verifier protocols do exist; they rely on post-quantum cryptography and interactive proofs [Mah18]. But they have their own trade-offs:

- *Prover overhead.* Preparing and measuring many more qubits per gate before any error correction as a consequence of the encryption of the qubits.
- *Security model.* Statistical security is replaced by security based on computational hardness assumptions.
- *Absence of Composability.* Remote State Preparation (RSP)—a crucial ingredient for delegated quantum computation—cannot be securely constructed under computational asumption in a composable framework [BCC$^+$20].

Hence a gap remains: a protocol with a quantum-minimal verifier yet that is still practical, statistically and composably secure.

Chapters 6 to 8 address this question by either:

- Reducing the number of states that need to be prepared by the verifier.
- Removing the need to prepare states, instead relying on states provided by the prover and only applying rotations on it.
- Replacing single-photon sources with weak-coherent-pulses generators.

3.4 Over-sensitivity to physical noise

Current VBQC schemes deliberately leave traps unencoded: a single-qubit phase-flip on a trap must reveal tampering. The strategy works against a malicious prover, but it backfires against ordinary device noise. As gate-level errors occur independently with probability p, a circuit of N gates therefore accumulates $O(pN)$ random phase flips—exactly the kind of events that an honest prover cannot avoid.

Because the protocols treat every unexpected trap outcome as dishonesty, the abort probability is overwhelming with circuit size. Proof-of-concept demonstrations survive, but a computation that stretches into the few-hundred-qubit or few-thousand-gate regime is likely to abort on nearly every run. The protocol scales in perfect qubit count—increasing the computation size does not demand more repetitions or larger logical encoding for a given targeted security error—, yet not when noise is taken in account. In short, known verification protocols would turn a possibly insecure noisy quantum computing device into a secure noisy quantum *non*-computing device!

In Chapter 10 we show how to remedy this fragility by delegating a fault-tolerant computation to the prover, thereby protecting traps and data alike while preserving verifiability and blindness.

3.5 Extending functionality

All existing blind-and-verified protocols assume a single verifier owns the input register. Practical workloads, however, often span several data owners who mistrust one another. This could be illustrated by hospitals collaboratively training in a quantum fashion a pattern recognition model over the entire population of a country. These scenarios call for Secure Multi-Party Quantum Computation (SMPQC), in which:

- Several clients each supply a disjoint subset of the inputs.
- A quantum server executes a joint algorithm.
- Every party learn only its authorized portion of the global output, and does not have access to the proprietary data of the other participants.

Yet, the existing MBQC-based SMPQC construction [KP17] has limited guarantees—blindness but not verifiability—and unusual requirements—absence of client-server collusion—, while such limitations do not exist for circuit-based SMPQC.

Chapter 9 introduces the first composable SMPQC protocol built atop the modular framework of Chapter 4.

Other extensions are currently being constructed but fall outside the scope of this document. They deal with benchmarking and noise characterization. Because they rely on the same techniques and broaden the range of applicability of the architecture that security imposes on hardware vendors, they further push them to put the necessary effort to incorporate into their roadmaps.

Part II
Modular and Composable Toolbox

This part introduces the toolbox [KKL+24] which played a crucial role in advancing verification techniques. It is a followup to [LMKO21], building upon the initial modularization capabilities it uncovered. This toolbox generalizes many of the earlier techniques, opening up new possibilities for optimization.

In the following sections, we will begin by deconstructing existing protocols—namely UBQC and VBQC—into their core modules. We will then demonstrate how these modules can be reassembled to form new, more efficient, and robust verification protocols. This will be illustrated through the robust VBQC protocol that laid the foundation of this work, as well as the dummyless protocol, which serves as a cornerstone for all subsequent optimizations presented in Parts III and IV.

4 Framework for Verification Protocols

This chapter, introduces a *modular* framework for designing verification protocols. Each module is assigned a single purpose that governs one aspect of the performance or adaptability of the whole protocol.

Dividing responsibilities in this way brings two immediate pay-offs:

- *Local reasoning.* Protocol security reduces to module-level properties checked in isolation. These properties are security for the subprotocol in charge of Remote State Preparation; detection, insensitivity, correctness for trappification; overhead for embedding.
- *Plug-and-play optimization.* One can refine a single module without reopening the entire security proof.

The rest of this chapter proceeds in two steps. First, Section 4.1 deconstructs UBQC (Protocol 2) and VBQC (Protocol 3), then each extracted module is reassembled into a template protocol (Protocol 4), for which only the instantiation of the various modules will be missing. Several instantiations of these modules are described in Parts III and IV, each tackling different challenges.

This instantiation is precisely the role of the next two parts of the document.

The work that motivated the construction of the framework presented here is [LMKO21]. [KKL+24] later distilled the crucial ingredients that were put to work and established the fully modular and composable form we adopt here.

4.1 Modularization

The task of the present section is simple to state: we isolate and formalize the three building blocks that recur in every verified and blind delegation scheme, each with a sharply defined role.

- *Remote State Preparation (RSP).* Enables blindness by allowing the verifier to initialize a single-qubit state inside the prover's device without disclosing its classical description.
- *Trappified Scheme.* Encapsulates deviation detection. The definition comes with quantitative notions of *detection*, *insensitivity*, and *correctness* that convert local error detection behavior into global security guarantees.
- *Embedding Algorithm.* Compiles an arbitrary target computation into the unspecified region of a graph state while respecting blindness and facilitating deviation detection amplification.

By the end of the section the reader will have:

- Rigorous resource or algorithm definitions for RSP, trappified canvases, the embedding algorithms.
- A clear view of how the three modules dovetail and which aspect of protocol efficiency or security each module governs.

Concrete protocols are deferred to the next section, where the modules are recombined into a template protocol and their local properties are lifted to full composable security.

The underlying idea behind verified-blind computation is relatively simple. It applies the "jealous couple technique" to get confidence on the answer to some question. It hides the true question into many other innocuous questions of the same kind but for which the answer is known. These will be traps. If all traps are passed, then answer to the true question is trusted. If one trap failed... One issue is clearly to ensure the potentially malicious party does not detect the traps by suspecting their nature—meaning trap questions cannot be reused and need to be as difficult as true questions. Another one is to boost the confidence quickly. This is impossible without error correction, which might be why jealous couples don't last long as life quickly turns into constant interrogation. Or maybe is it because of absence of noise robustness in the provided answers.

Extracted Functionalities from VBQC Protocols

The Verifiable Blind Quantum Computing (VBQC) constructions—from the original one [FK17] to the dotted-triple-graph VBQC (Protocol 3)—rest on three logical ingredients, each of which can be isolated and studied on its own.

- *Blind execution of a computation class.* UBQC (Protocol 2) allows the verifier to steer the prover through any BQP computation that fits within a given

graph state, simply by adjusting single-qubit measurement angles and inserting dummies to break edges. Crucially, when those angles are restricted to multiples of $\pi/2$ the unitary describing the computation falls inside the Clifford group and becomes classically simulable. The same interaction pattern between verifier and prover therefore supports both interesting quantum instances and deterministic, classically easy-to-predict ones. Yet, delegated through UBQC, easy instances remain indistinguishable form genuinely quantum ones to the prover.

- *Local probes of malicious behavior.* A single-qubit disconnected from the graph—or, more generally, a small deterministic computation embedded in the same resource state—can serve as a *trap*. Any deviation that flips the outcome of the trap signals cheating. Because not every qubit can be sacrificed for making traps—otherwise no qubit would remain to carry on the computation of the verifier—traps must be sprinkled probabilistically, and their locations kept hidden from the prover.
- *Amplification of detection probability.* Because traps are chosen probabilistically and because for each trap some deviations are never detected, simply inserting traps is not enough to provide negligible security error. VBQC boosts the detection probability by encoding the verifier's data across many physical qubits—either via a fault-tolerant scheme or by classical repetition—and then distributing traps throughout that enlarged structure. A deviation, strong enough to alter the logical outcome, must now disturb many sites. These higher-weight *harmful* deviations trigger at least one trap with overwhelming probability.

These three functions—blind state preparation and control, trap creation and evaluation, and amplification by fault-tolerance or repetition—will be detailed as the *Remote State Preparation*, *Trappification*, and *Embedding* modules defined formally in the subsections that follow.

Blindness: Remote State Preparation

Motivation

The origin of blindness in UBQC—the functionality that restores the balance of powers between the weak verifier and powerful prover—is simple to state: the prover must hold a qubit whose classical description he does not know. In the two-prover variant of UBQC [BFK09], this role is played by the second prover, which supplies the qubits on the verifier's behalf. Lo's early work on remote preparation [Loo0] and its cryptographic formalization [DKL12] make clear that this essential functionality can be abstracted away from the rest of the protocol. It is commonly captured in a resource called *Remote State Preparation (RSP)*.

The practical requirement is modest—prepare either $|+_\theta\rangle$ for $\theta \in \Theta$ or a computational basis state $|0\rangle, |1\rangle$. Yet this single primitive underlies UBQC, VBQC, and the protocols developed later in this document.

Definition

These requirements can be turned into a proper resource within the abstract cryptography framework (Chapter A) where a sender selects a desired state to be prepared at a distant receiver's interface. With Alice being the sender and Bob the receiver, RSP can be defined in the following way:

Resource 3. Remote State Preparation

> **Inputs:** Alice inputs a symbol $\lambda \in \Lambda$.
> **Computation by the Resource:** The Resource prepares the state $|\lambda\rangle$ at Bob's interface. The alphabet Λ is either the equatorial set $\{+_\theta : \theta \in \Theta\}$ or that set union $\{0, 1\}$ when computational-basis dummies are required. In the former case, we call the resource *single-plane RSP*.

The definition is deliberately spare. No side-channel is exposed beyond the quantum state itself. The qubit is guaranteed correct *at the moment of delivery*. What

happens afterwards—unitary drift, stochastic noise, measurement, or arbitrary deviation inserted by the prover—must be independent of λ. This makes it *secure-by-design* as it is simple enough to check the definition fits its textual description.

In fact, secure-by-design means that a definition is promoted to a security specification.

Implementation issues

Implementing such ideal RSP in hardware is subtle. Imperfect sources or mistimed classical control can imprint a faint λ-dependent signature on auxiliary degrees of freedom—timing, phase noise, or photon number. Such leakage will break blindness and invalidate composable proofs. Later chapters return to this point:

- Chapter 7 shows that a single trusted $\mathsf{Z}(\theta)$ rotation, combined with X bit flips, suffices for VBQC with minimal overhead for the verifier.
- Chapter 8 presents a composable RSP protocol that tolerates weak coherent pulse sources instead of single photon sources while maintaining the λ-independence demanded by Resource 3.

Deviation Detection: Trappification

Motivation

The original VBQC protocols use the simplest possible probe: a single-qubit, prepared as $|+\rangle$ and disconnected from the rest of the graph, whose X-measurement outcome must be 0. Any non-trivial Z-type deviation on that site flips the bit and is caught. This one-qubit trap is easy to analyze and easy to hide, but it is also extremely narrow: it only tests a tiny corner of the prover's behavior at a time.

Two further observations drive the present generalization. (i) Not every qubit can be a trap. Some must carry the actual computation. Traps therefore have to be sprinkled sparsely and at random, with their locations hidden by blindness; otherwise the prover could simply tip-toe around them. (ii) What truly matters is not the size of the trap but its predictability. A trap can be any deterministic, classically easy-to-evaluate MBQC pattern run on a subgraph and whose outcome the verifier can compute offline. Once this is recognized, larger traps are admissible. A first example of such a multi-qubit trap already appeared in [FKD18]. Here we systematize the idea and use it to shape the detection profile of the overall scheme.

This broader view pays off twice. It gives far more latitude in how and where traps are inserted, and it lets us tailor their detection profile to the amplification strategy used later on. The formal objects—partial patterns, trappified canvases, and trappified schemes—introduced in the next subsection make that flexibility precise.

Definitions

Our goal is clear: traps must sit beside the computation, yet still detect any deviation that could alter it. If the computation always uses the same physical vertices, an adversary can target those and hide behind legitimate measurement instructions to deviate. We therefore need to randomize where the computation lives and where traps sit. Formally capturing that idea requires three ingredients:

1. A partially specified MBQC pattern, so we can leave room for either or both trap and computation.
2. A trappified canvas that bundles such a partial pattern together with its input state and an acceptance rule which together allow deviation detection.
3. A trappified scheme: a distribution over many trappified canvases, all indistinguishable to the prover, plus an embedding algorithm that plugs the actual computation into the blank spaces.

We now introduce these notions, keeping the intuition in view.

Definition 2. Partial MBQC Pattern

Given a graph $G = (V, E)$, a partial pattern P on G is defined by:

- $G_P = (V_P, E_P = E \cap V_P \times V_P)$, a subgraph of G;
- 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;
- $\{\phi(i)\}_{i\in V_P\setminus O_P}$, a set of measurement angles;
- $f_p : V_P \setminus O_P \to V_P \setminus I_P$, a flow inducing a partial order $\preceq_P$ on V_P.

Think of a partial pattern as an MBQC pattern where some measurement angles are fixed, others are left blank; some input and output vertices are labeled, others are still free. This lets us say this contains small sub-computations—a potential trap—without committing the rest of the pattern yet, save for the graph itself and order of measurement.

Definition 3. Trappified Canvas

A *trappified canvas* $(T, \sigma, \mathcal{T}, \tau)$ on a graph $G = (V, E)$ consists of:
- T, a partial pattern on a subset of vertices V_T of G with input and output sets I_T and O_T;
- σ, a tensor product of single-qubit states on $\Pi_{I,T}$;
- $\mathcal{T}$, an efficiently classically computable probability distribution over binary strings[3];
- 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 aborts for $\tau(t) = 1$.

[3] Here, we really mean computing the probability distribution, not just sampling from it. This is beacuse the decision algorithm τ might need to access $\mathcal{T}$ while being executed. In practice, $\mathcal{T}$ is so that, given T, a single output string has probability 1, the rest being equal to 0, and identification of the only likely outcome is classically easy.

A trappified canvas is a self-contained trap gadget: it specifies what to prepare, what to measure, how to post-process the outcomes, and how to decide accept or abort, while still having unaffected space available for computing. Compared with the original single-qubit trap, this allows multi-qubit, deterministic sub-computations—as long as the verifier can efficiently predict $\tau(t)$ in advance.

Definition 4. Trappified Pattern

Given a computation $\mathsf{C} \in \mathfrak{C}$ and a trappified canvas T on graph G with order $\preceq_G$, we call the completed pattern $C \cup T$ which computes C over the unaffected space of the canvas a *trappified pattern*.

The canvas leaves white space that is filled with the real computation C. We will see later, that there are usually many possibilities to embed a computation. The choice of embedding algorithm $E_{\mathfrak{C}}$ will be discussed later as it governs the detection amplification.

As different runs choose different canvases, the prover must not be able to tell where the logical data ended up. This requires an extra care about blindness. If two canvases differed in their graph, the prover could recognize which canvas was used and, by elimination, where traps sit. We therefore need:

Definition 5. 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.

In short, blind-compatibility ensures that all the trappified patterns share the same graph and the same scheduling for their measurements. Only the hidden angles and state choices differ, so blindness is preserved.

Finally, we package everything:

Definition 6. 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:
- $\boldsymbol{P}$, a set of *blind-compatible* trappified canvases over graph G with common partial order $\preceq_{\boldsymbol{P}}$.

- $\preceq_G$, a partial ordering of vertices V of G that is compatible with $\preceq_P$.
- $\mathcal{P}$, a probability distribution over the set $\boldsymbol{P}$ which can be sampled efficiently.
- $E_{\mathfrak{C}}$, a *proper* embedding algorithm for $\mathfrak{C}$.

The scheme is the hat from which the verifier draws a canvas at random, then embeds the computation. The embedding algorithm $E_{\mathfrak{C}}$ (deferred to Definition 10) ensures that there is a uniform way to map the computation C into the unaffected space of the trappified canvas and properness (deferred to Definition 11) prevents information from flowing from the logical computation into the traps, ensuring the preservation of blindness and enabling a composable proof.

Properties

A trappified scheme by itself is only raw material. To turn it into a verification protocol we must quantify three things (See Fig. 4.1):

1. How often do we cry wolf when truly harmful deviation happened (*detection*).
2. How often do we cry wolf when nothing important was touched (*insensitivity*).
3. If we do not cry wolf, how close is the logical output to what it should have been? (*correctness*).

Formally, the random Pauli encryption in UBQC turns an arbitrary channel into a Pauli channel; see, e.g., the standard twirling lemma [DCEL09, Kap16]. As result, any adversarial CPTP map imposed by the prover reduces to a convex combination of Pauli operators. From this point on it thus suffices to analyze Pauli deviations.

The first property we quantify is the Pauli detection capability of the scheme relative to a set of deviations $\mathcal{E}$. This set is meant to be the set of harmful deviations: [4]

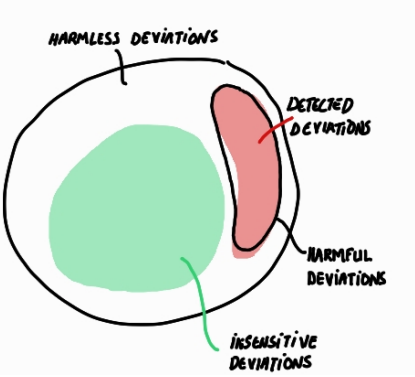


Figure 4.1: Picturing the various types of deviations and their impact on the computation. We want to quantify how often we don't detect deviations that can change the result of the computation *at the logical level*. When this is very small, we will have a verification scheme with good security.

Definition 7. Pauli Detection

Let T be a trappified canvas sampling from distribution $\mathcal{T}$. Let $\mathcal{E}$ be a subset of the Pauli group $\mathcal{P}_V$ over the graph vertex qubits. For $\epsilon > 0$, we say that T *ϵ-detects* $\mathcal{E}$ if:

$$\forall \mathsf{E} \in \mathcal{E},\ \Pr_{t \leftarrow\$ \mathcal{T}_{\mathsf{E}}}[\tau(t) = 1] \geq 1 - \epsilon. \tag{4.1}$$

We say that a trappified scheme $\boldsymbol{P}$ *ϵ-detects* $\mathcal{E}$ if:

$$\forall \mathsf{E} \in \mathcal{E},\ \sum_{T \in \boldsymbol{P}} \Pr_{\substack{T \leftarrow\$ \mathcal{P} \\ t \leftarrow\$ \mathcal{T}_{\mathsf{E}}}}[\tau(t) = 1, T] \geq 1 - \epsilon. \tag{4.2}$$

There are also deviations that we do not manage to detect. This is not always detrimental as long as they have little to no impact on the result of the computation.

Definition 8. Pauli Insensitivity

Let T be a trappified canvas sampling from distribution $\mathcal{T}$. Let $\mathcal{E}$ be a subset of $\mathcal{P}_V$. For $\delta > 0$, we say that T is *δ-insensitive to* $\mathcal{E}$ if:

$$\forall \mathsf{E} \in \mathcal{E},\ \Pr_{t \leftarrow\$ \mathcal{T}_{\mathsf{E}}}[\tau(t) = 0] \geq 1 - \delta. \tag{4.3}$$

We say that a trappified scheme $\boldsymbol{P}$ is *δ-insensitive to* $\mathcal{E}$ if:

$$\forall \mathsf{E} \in \mathcal{E},\ \sum_{T \in \boldsymbol{P}} \Pr_{\substack{T \leftarrow\$ \mathcal{P} \\ t \leftarrow\$ \mathcal{T}_{\mathsf{E}}}}[\tau(t) = 0, T] \geq 1 - \delta. \tag{4.4}$$

And finally, we want to capture the distance between the ideal computation and what the verifier actually gets when a deviation is sampled from a given set and no trap fires.

Definition 9. Pauli Correctness

[4] In Eq. (4.1) the notation $t \leftarrow\$ \mathcal{T}_{\mathsf{E}}$ refers to the sampling of t according to the trap measurement when the trap T is affected by deviation E applied after the correct unitary part of the trap applied. The same notation is used in the subsequent equations.

Indeed, we will follow this intuition later in order to obtain global robustness.

Let $(T, \sigma, \mathcal{T}, \tau)$ be a trappified canvas on graph G, $\preceq_G$ an order on the vertices of G and $E_{\mathfrak{C}}$ an embedding algorithm. Let $C \cup T$ be the trappified pattern obtained by embedding a computation $\mathsf{C} \in \mathfrak{C}$ on T using $E_{\mathfrak{C}}$ and order $\preceq_G$. Let I_C be the set of input vertices for the computation C in $C \cup T$ and let $|\psi\rangle$ be a state on $|I_C| + |R|$ qubits, for sufficiently large auxiliary system R, such that $\mathrm{Tr}_R(|\psi\rangle) \in \Pi_{I,\mathfrak{C}}$, where $\Pi_{I,\mathfrak{C}}$ is the verifier's input subspace. Let $\mathcal{E}$ be a subset of $\mathcal{P}_V$. For $\mathsf{E} \in \mathcal{E}$, we define $\tilde{\mathsf{C}}_{T,\mathsf{E}} = \mathsf{D}_{O,\mathfrak{C}} \circ \mathrm{Tr}_{O_C^c} \circ \mathsf{E} \circ (C \cup T)$ to be the CPTP map resulting from applying the trappified pattern $C \cup T$ followed by $\mathsf{D}_{O,\mathfrak{C}}$, the decoding algorithm that maps the output space $\Pi_{O,\mathfrak{C}}$ to regular qubits. For $\nu \geq 0$, we say that T is *ν-correct* on $\mathcal{E}$ if:[5]

$$\forall \mathsf{E} \in \mathcal{E},\ \forall \mathsf{C} \in \mathfrak{C},$$
$$\max_{\psi} \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C} \otimes \mathsf{I}_T) \otimes \mathsf{I}_R[|\psi\rangle\langle\psi| \otimes \sigma]\|_{\mathrm{Tr}} \leq \nu. \tag{4.5}$$

This is extended to a trappified scheme $\boldsymbol{P}$ by requiring the bound to hold on average over $T \in \boldsymbol{P}$:

$$\forall \mathsf{E} \in \mathcal{E},\ \forall \mathsf{C} \in \mathfrak{C},$$
$$\max_{\psi} \left(\sum_{T \in \boldsymbol{P}} \Pr_{T \leftarrow \$ \mathcal{P}}[T] \|(\tilde{\mathsf{C}}_{T,\mathsf{E}} - \mathsf{C} \otimes \mathsf{I}_T) \otimes \mathsf{I}_R[|\psi\rangle\langle\psi| \otimes \sigma]\|_{\mathrm{Tr}} \right) \leq \nu. \tag{4.6}$$

[5] Note that in Eq. (4.5) we are simply expanding the diamond norm between the correct and deviated CPTP maps, but as we need to still fix the input subspace and the input for the trap qubits, this expression avoids defining the deviated CPTP map on the logical result directly.

Detection and insensitivity are complementary knobs, yet they characterize different sets of deviations. Correctness then ties everything together:

- If a deviation escapes detection, it must nevertheless leave the logical result almost untouched.
- Conversely to obtain robustness, traps must be insensitive to deviations that almost don't touch the result.

Detection Amplification: Embedding

Motivation

Traps by themselves only give an inverse–polynomial chance of catching a cheat. Detection amplification comes from spreading the logical computation across a larger structure—either into a fault–tolerant error-correcting scheme code or into several independent repetitions—and sprinkling traps through that structure. The embedding algorithm is the classical piece of machinery that performs this spread:

- It maps the target computation into the unused area of the chosen canvas.
- It fixes how logical outputs are decoded (majority vote, code decoder,...).
- It ensures that small, local Pauli errors are *harmless* (corrected or outvoted), while large, *harmful* ones are likely to hit a trap.

Abstract Cryptography forces one more constraint: the algorithm must be public and must not let information flow from the logical computation back into the traps. Otherwise blindness, and hence the simulator in the security proof, would fail as the choice of computation could influence the trap detection probability for a given deviation. This is the content of *properness* below. This is the analog of the independent local-verifiability criterion used in [DFPR14]. Here, the property appears very explicitly as a necessary ingredient used to construct the simulator in the security proof.

Definitions

Definition 10. 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:

- $\mathsf{C} \in \mathfrak{C}$, the computation to be embedded.
- $G = (V, E)$, a graph, and an output set O.
- T, a trappified canvas on graph G.

- $\preceq_G$, a partial order on V which is compatible with the partial order defined by T.

and outputs:

- A partial pattern C on $V \setminus V_T$, with
  - Input and output vertices $I_C \subset V \setminus V_T$ and $O_C = O \setminus O_T$.
  - Two subspaces (resp.) $\Pi_{I,\mathfrak{C}}$ and $\Pi_{O,\mathfrak{C}}$ of (resp.) $I_{\mathfrak{C}}$ and $O_{\mathfrak{C}}$ with bases (resp.) $\mathcal{B}_{I,\mathfrak{C}}$ and $\mathcal{B}_{O,\mathfrak{C}}$.
- A decoding algorithm $\mathsf{D}_{O,\mathfrak{C}}$.

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 $\perp$.[6]

[6] This is required to ensure that the behavior of the embedding algorithm is well defined for all possible input computation C. This is important to cover the case where the distinguisher could try to input very large computations that cannot fit into the available blank space in the trappified canvas.

In this definition, $E_{\mathfrak{C}}$ fills the blanks of the canvas with the computation pattern and tells the verifier how to read the answer back. Different choices of $E_{\mathfrak{C}}$ yield different amplification behaviors.

If the embedding is such that it stays in the logical space of a quantum error-correcting code, deviations with a weight smaller than half the minimum distance of the code will be harmless. The set of deviations that would need to be detected are those that are not low weight under this criterion, and that would typically be done with a very high probability, even on average over the choice of trappified canvas. Similarly, whenever the computation is classically repeated and the result obtained by majority vote, we are computing in a classically protected subspace. This will give the correct result as long as there are less than half the rounds that are affected by a deviation. There again we will be able to isolate the low-weight from high-weight deviations, the latter being detected with very high probability.

With the embedding comes the subtle notion of *proper embedding* which captures the absence of influence between the computation and the traps.

Definition 11. 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:

- The flow f does not induce dependencies on vertices V_T of partial pattern T, in the sense of Eq. (2.1).
- The input and output subspaces $\Pi_{I,\mathfrak{C}}$ and $\Pi_{O,\mathfrak{C}}$ do not depend on the trappified canvas T save for a relabelling of the qubits.

In essence, a proper embedding guarantees that the simulator in the security proof can sample a trappified canvas and any computation from the class $\mathfrak{C}$ and play the role of the verifier with the distinguisher playing that of the prover. The trap would be computed and yield the same transcript even if the actual computations would differ. Without the above property, this could not be guaranteed and the distinguisher could use it to its advantage.

4.2 Reconstruction

The previous section carved verification into three modules—RSP for blindness, Trappification for deviation detection, Embedding for amplification. Here we put the pieces back together. The goal is twofold: (i) give a concrete, VBQC-style protocol template that calls each module; (ii) show how local guarantees—detection, insensitivity, correctness—combined with the chosen amplification strategy lift to a global, composable security bound. This reconstruction also makes plain how each module's parameters feed directly into the overall performance.

Template protocol

An informal template for VBQC-style protocols using the notions defined earlier can be described succinctly, with Alice being the verifier and Bob the prover:

Protocol 4. Trappified Blind Delegated Quantum Computation, Informal

1. Alice samples a trappified canvas from the trappified scheme and embeds its computation in the canvas, yielding a trappified pattern.
2. Alice blindly delegates this trappified pattern to Bob via UBQC, and collects the output qubits and the measurement outcomes.
3. Alice evaluates the decision function, and accepts or aborts.
4. If she didn't abort, Alice performs some simple classical or quantum single-qubit post-processing on the output.

If we restrict to BQP computations, both inputs and outputs are classical and the amplification can be performed through repetitions and majority voting. This way, we arrive at a more concrete template for verified BQP computations, where only traps and RSP implementations are left to specify:

Protocol 5. Trappified Blind Delegated Quantum Computation

Public Information:
- $G = (V, E, I, O)$, a graph with input and output vertices I and O respectively.
- $\boldsymbol{P}$, a trappified scheme on graph G.
- $\preceq_G$, a partial order on the set V of vertices.
- N, d, w, parameters representing the number of runs, the number of computation runs, and the number of tolerated failed tests.

Alices's Inputs: A set of angles $\{\phi_i\}_{i\in V}$ and a flow f which induces an ordering compatible with $\preceq_G$.

Protocol:
1. Alice samples uniformly at random a subset $C \subset [N]$ of size d representing the runs which will be its desired computation, henceforth called computation runs.
2. For $k \in [N]$, Alice and Bob perform the following:
   (a) If $k \in C$, Alice sets the computation for the run to its desired computation $(\{\phi_i\}_{i\in V}, f)$. Otherwise, Alice samples a test (T, σ, τ) from the trappified scheme $\boldsymbol{P}$.
   (b) Alice and Bob blindly execute the run using the UBQC.
   (c) If it is a test, Alice uses τ on the measurement results to decide whether the test passed or not.
3. At the end of all runs, let x be the number of failed tests. If $x \geq w$, Alice aborts and outputs $(\perp, \mathsf{Abort})$.
4. Otherwise, Alice accepts the computation. It then performs a majority vote on the output results of the computation runs and sets the result as its output.

Key Theorems

The following theorems can be applied to protocols following the informal template and are given here only with a brief sketch of proof. Their proofs nonetheless follow an intuitive route:

- UBQC turns any deviation into a mixture of Paulis.
- Split that set into detected, ignored but harmless, and the rest.
- Bound each contribution to the distinguishing advantage.

Theorem 3. Detection Implies Verifiability, Informal

Let $\mathcal{E}_\epsilon, \mathcal{E}_\nu \subset \mathcal{P}_V$, where $\mathcal{P}_V$ is the set of Pauli operators on the qubits indexed by the graph vertices (deviations), such that:
- $\mathcal{P}_V \setminus \mathcal{E}_\epsilon \subseteq \mathcal{E}_\nu$.
- $\mathsf{I} \in \mathcal{E}_\nu$.

If Trappified Blind Delegated Quantum Computation uses a trappified scheme which:
- *ϵ-detects $\mathcal{E}_\epsilon$.*
- *is δ-insensitive to at least $\{\mathsf{I}\}$.*
- *is ν-correct on $\mathcal{E}_\nu$.*

then the protocol is $\max(\epsilon, \delta + \nu)$-secure against an arbitrarily malicious unbounded prover.

Sketch of Proof. In short: (i) with probability at most ϵ a harmful Pauli slips past detection; (ii) undetected Paulis lie in $\mathcal{E}_\nu$, so their logical effect is bounded by ν; (iii) insensitivity ensures we do not falsely abort more than δ. Triangle inequality gives the stated bound. ■

The following theorem addresses the sensitivity to noise of the initial VBQC protocols [FK17, KW17]. It links insensitivity to robustness:

Theorem 4. Robust Detection Implies Robust Verifiability (Informal))

We assume now that the prover in Trappified Blind Delegated Quantum Computation is honest-but-noisy: the error applied is in $\mathcal{E}_\delta$, the set of δ-insensitive deviations, with probability $(1-p_\delta)$ and $\mathcal{P}_V \setminus \mathcal{E}_\delta$ with probability p_δ. Then, the verifier accepts with probability at least $(1-p_\delta)(1-\delta)$. If furthermore $\mathcal{E}_\delta \subseteq \mathcal{E}_\nu$, the set of ν-correct deviations, then the total correctness error is $p_\delta + \delta + \nu$.

Sktech of Proof. The theorem is obtained after quantifying: (i) how often honest-noise falls in the insensitive set, passes the tests and gives an incorrect result; and (ii) how often it falls outside of it and is then likely to provide an erroneous result. ■

In essence, it acknowledges that by tuning the acceptance function, we can change the size of the set of deviations that the trappified scheme is insensitive to. By choosing the embedding accordingly, this allows to minimize the size of the deviations that yield a correct result but are detected and therefore make the protocol abort. By making these adjustments, it is possible to ensure that the insensitive set is as large as possible and comprises the deviations that are due to noise with high probability.

Combining both, we get the concrete, BQP-specific statement:

Theorem 5. Security and Noise-Robustness of Protocol 4, Theorem 13 from [KKL+24]

Let BQP_G be the set of BQP computations that can be performed on graph G, with bounded error c. Let $\boldsymbol{P}$ be a trappified scheme on graph G and let $\mathcal{E}_\epsilon, \mathcal{E}_\delta, \mathcal{E}_\nu$ be subsets of Pauli deviations such that:

- $\mathcal{E}_\nu \subseteq \{\mathsf{E} \in \mathcal{P}_V \mid \forall \mathsf{C} \in \mathsf{BQP}_G, \forall T \in \boldsymbol{P}, \tilde{\mathsf{C}}_{T,\mathsf{E}} = \tilde{\mathsf{C}}_{T,\mathsf{I}}\}$.
- $\mathcal{P}_V \setminus \mathcal{E}_\epsilon \subset \mathcal{E}_\nu$.
- *$\boldsymbol{P}$ ϵ-detects $\mathcal{E}_\epsilon$, is δ-insensitive to $\mathcal{E}_\delta$ and perfectly insensitive to I.*
- *The honest prover's noise is modelled by sampling for each computation or test round an error $\mathsf{E} \in \mathcal{E}_\delta$ with probability p_δ and $\mathsf{E} = \mathsf{I}$ with probability $1-p_\delta$.*

Let n, d, w be defined as in Protocol 5. If the following conditions are satisfied:

- *$\frac{w}{(n-d)(1-\epsilon)}$ is away by a constant and upper-bounded by $\frac{1-2c}{2-2c}$.*
- *p_δ is away by a constant and upper-bounded by $\frac{w}{(n-d)\delta}$.*

Then Protocol 4, using $\boldsymbol{P}$, $\eta(n)$-constructs the Secure Delegated Quantum Computation (Resource 2) for computations in set BQP_G in the Abstract Cryptography framework.

Note that the value of $\eta(n)$ heavily depends on the value of δ and ϵ, in particular via the coefficient in the exponential. This is a motivation for carefully designing the trappification scheme and embedding algorithm.

Traps from Stabilizer Tests

To instantiate a rich family of trappified canvases on any graph state, subset-stabilizer tests are natural. Let S be the stabilizer group for $|G\rangle\langle G|$, the graph state associated to $G = (V, E)$, and consider the set of canonical generators of S equal to $\{\mathsf{S}_v = \mathsf{X}_v \bigotimes_{(v,w)\in E} \mathsf{Z}_w, v \in V\}$. We can write uniquely any $\mathsf{R} \in \mathsf{S}$ as $\prod_{v\in V} S_v^{r_v}$ with $r_v \in \{0,1\}$ and define $\mathbb{1}_\mathsf{R} := \{v \in V \mid r_v = 1\}$ to be the support of R in the generators S_v.

A test round for $\mathsf{R} \in \mathsf{S}$ can be obtained by:

1. Input: prepare an eigenstate of R.
2. Compute: Instruct an MBQC pattern consisting of measuring R.
3. Decide: Accept iff the measured outcome corresponds to the prepared eigenstate.

The test for a given stabilizer $\mathsf{R} \in \mathsf{S}$ is defined by having the verifier prepare each qubit v in the state:

- $|0\rangle$ if $\mathsf{R}(v) = \pm\mathsf{Z}$.
- $|+\rangle$ otherwise.
- It flips the qubit—while staying in the same basis—for v_0 if there was a minus sign in R.

The verifier then simply instructs the prover to measure each qubit in the Y-basis if $\mathsf{R}(v) = \pm\mathsf{Y}$ and in the X basis otherwise, getting outcome $t(v)$ for vertex v. It then computes $\tau(t) := \bigoplus_{v \in \mathbb{1}_\mathsf{R}} t(v)$ to determine the outcome of the measurement of R on the inputs provided by the verifier. For an honest prover, this value should return 0.

The freedom in choosing which stabilizers R to include in the trappified scheme $\boldsymbol{P}$ will be at the core of constructing dummyless verification protocol.

4.3 Takeaways

We have split verifiable blind delegation into three minimal modules (RSP, Trappification, Embedding) and then reassembled them into a full protocol with composable guarantees. The broader lessons are:

- *Modularization is leverage, not ornament.* Once blindness, deviation detection, and amplification are isolated, security reduces to checking a short list of local properties (detection, insensitivity, correctness) plus an explicit amplification rule. Proof effort scales with the module you touch, not with the whole protocol.
- *Blindness is a primitive.* Treating Remote State Preparation as its own resource clarifies what must remain hidden (the classical label of the prepared state) and what may leak (nothing beyond the state itself). It also makes room for alternative implementations (two-prover, weak coherent pulses, limited rotations) without disturbing the rest of the stack.
- *Traps are computations, not just qubits.* Starting from single-qubit traps, we generalized the concept to trappified canvases and schemes. Any deterministic, classically predictable partial pattern qualifies, provided its outputs can be evaluated from MBQC outcomes.
- *Proper embedding is the hinge between detection and amplification.* It must (i) keep information from the computation from flowing into the traps, to preserve blindness and enable simulation, and (ii) place the computation inside a redundancy structure—code space or repetition set—so that harmful deviations are necessarily high weight. Low-weight deviations are then corrected and should fall into the insensitivity set.
- *Local parameters drive global bounds.* The trio (detection, insensitivity, correctness) and the amplification policy (code distance, number of repetitions, acceptance threshold) map directly to the security error in Abstract Cryptography and to the robustness of the protocol. Tuning any of them is a transparent trade-off: stricter detection raises protection against cheating but can hurt robustness to honest noise, and vice versa.

A design workflow also emerged:

1. Pick or engineer an RSP implementation that matches the verifier's hardware.
2. Choose a trappified scheme that detects the deviations you care about and is insensitive to the noise you expect.
3. Select an embedding/amplification strategy that makes undetected harmful attacks improbable.
4. Read off the global bounds from the template theorems.

In the next chapters we exploit this plug-and-play structure: we swap the naive RSP implementation for a lean one to lighten the verifier, design dummyless trappification to cut prover overhead, extend the orchestration to multi-party settings, and finally address verifier-side noise robustness. Each improvement touches one module (or its interface) while leaving the others intact.

5 Robust VBQC

We now take the modular machinery for a spin. This chapter instantiates the three modules of the previous one—Remote State Preparation for blindness, Trappification for deviation detection, and Embedding for amplification—in the simplest non–fault-tolerant setting. The target class is BQP, so amplification comes from plain repetition and a majority vote, not from a bulky quantum code. Tests and computations are pulled apart into separate rounds: traps live in their own, and a single-round looks no heavier than the unprotected UBQC pattern.

The objectives are:

- Show, concretely, how a full VBQC-style protocol is instantiated once the three modules are fixed.
- Drive the prover's space overhead to zero: every qubit in a computation round is used for the computation, while traps are exiled to distinct test rounds.
- Tune the acceptance rule so that honest physical noise—at moderate rate—is treated as harmless, giving a first notion of robustness without fault-tolerance.
- Read off global, composable security bounds directly from the local properties—detection, insensitivity, correctness—established for trappification and using the chosen amplification.

In short, this chapter is an example, not a detour: it shows that the framework is practical and that meaningful robustness and low overhead can already be achieved with repetition alone, provided one is content with classical outputs.

5.1 Protocol

The construction below is the straight-line instantiation of our three modules for the special case of BQP computations. Two design choices drive everything:

- *Amplification by repetition only.* No fault-tolerant code is inserted; we just repeat the whole delegated computation d times and take a majority vote.
- *Zero space overhead on the prover.* Computation and tests live in different rounds. In a computation round every qubit is used for the target MBQC pattern; in a test round every qubit serves the trap. Thus a single honest computation round costs exactly what an unprotected UBQC run would cost.

Security is therefore tied to how many rounds we run, not to inflating a single-round. This will let us tolerate a moderate amount of honest noise by loosening the trap acceptance threshold.

For a given graph G with flow f, that implicitely defines a computation class $\mathfrak{C}$, we denote by $\{V_i\}_{i\in K}$ a $|K|$-coloring of G. With that in place, the following protocol verifies computations in $\mathfrak{C}$ (See Fig. 5.1 for a visual definition of the various types of rounds used by the protocol).

Protocol 6. Robust VBQC

Alice's Inputs: Angles $\{\phi_v\}_{v\in V}$ and flow f on graph G, classical input to the computation $x \in \{0,1\}^{|I|}$.

Protocol:

1. Alice chooses uniformly at random a partition (C, T) of $[n]$ $(C \cap T = \emptyset)$ with $|C| = d$, the sets of indices of the computation and test rounds respectively.
2. For $j \in [n]$, Alice and Bob perform the following sub-protocol (Alice may

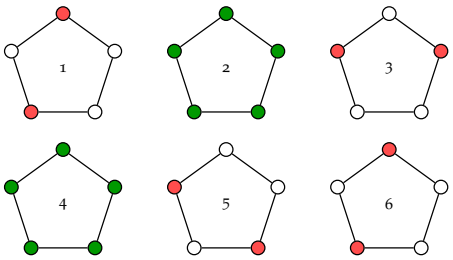


Figure 5.1: Possible sequence of 6 rounds of computation and trap rounds for a 5-qubit graph MBQC. Green qubits are used for computation, Red ones are for traps, while white ones are for dummies isolating single-qubit traps.

send message Redo_j to Bob before step 2.c while Bob may send it to Alice at any time, both parties then restart round j with fresh randomness):

(a) If $j \in T$ (test), Alice chooses uniformly at random a color $\mathsf{V}_j \leftarrow\$ \{V_k\}_{k\in[K]}$ (this is the set of traps for this test round).

(b) Alice sends $|V|$ qubits to Bob. If $j \in T$ and the destination qubit $v \notin \mathsf{V}_j$ is a non-trap qubit (therefore a dummy), then Alice chooses uniformly at random $d_v \leftarrow\$ \{0,1\}$ and sends the state $|d_v\rangle$. Otherwise, Alice chooses at random $\theta_v \leftarrow\$ \Theta$ and sends the state $|+_{\theta_v}\rangle$.

(c) Bob performs a CZ gate between all its qubits corresponding to an edge in the set E.

(d) For $v \in V$, Alice sends a measurement angle δ_v, Bob measures the appropriate corresponding qubit in the δ_v-basis, returning outcome b_v to Alice. The angle δ_v is defined as follows:

- If $j \in C$ (computation), it is the same as in UBQC, computed using the flow and the computation angles $\{\phi_v\}_{v\in V}$. For $v \in I$ (input qubit) Alice uses $\tilde{\theta}_v = \theta_v + x_v\pi$ in the computation of δ_v.
- If $j \in T$ (test): if $v \notin \mathsf{V}_j$ (dummy qubit), Alice chooses it uniformly at random from Θ; if $v \in \mathsf{V}_j$ (trap qubit), it chooses uniformly at random $r_v \leftarrow\$ \{0,1\}$ and sets $\delta_v = \theta_v + r_v\pi$.

3. For all $j \in T$ (test round) and $v \in \mathsf{V}_j$ (traps), Alice verifies that $b_v = r_v \oplus d_v$, where $d_v = \bigoplus_{i\in N_G(v)} d_i$ is the sum over the values of neighboring dummies of qubit v. Let c_{fail} be the number of failed test rounds (where at least one trap qubit does not satisfy the relation above), if $c_{fail} \geq w$ then Alice aborts by sending message Abort to Bob.
4. Otherwise, let y_j for $j \in C$ be the classical output of computation round j (after corrections from measurement results). Alice checks whether there exists some output value y such that $|\{y_j \,|\, j \in C, y_j = y\}| > \frac{d}{2}$. If such a value y exists (this is then the majority output), it sets it as its output and sends message Ok to Bob. Otherwise it sends message Abort to Bob.

5.2 Module definitions

Remote State Preparation

The RSP implementation is the naive one. We simply have the verifier prepare quantum states that are sent to the prover via a quantum channel. This is enough for now as any post-delivery noise is by construction *θ-independent*. With Alice playing the verifier and Bob the prover we have:

Protocol 7. RSP (Naive Instantiation)

Public Information: $\Lambda = \{0, 1, +_\theta \text{ for } \theta \in \Theta\}$ the classical description of the states that can be prepared.
Alice's Input: $\lambda \in \Lambda$
Preparation of the state:
- Alice prepares $|\lambda\rangle$ in a quantum register.

Transmission of the state:
- Alice sends the quantum register prepared in $|\lambda\rangle$ to Bob.

Trappification

Because amplification is done using time in sequential rounds, not in space, the trappified canvases are simple: the trap is made of extra copies of the same graph state $|G\rangle$ prepared with inputs that force deterministic outcomes (See Fig. 5.1).

- For a computation round: inputs and angles follow UBQC for the computation defined by G, flow f and measurement angles ϕ_v, $v \in V$.
- For a test round: we prepare another copy of G, but (i) fill non-trap vertices with dummies $|0\rangle, |1\rangle$ to isolate single trap-qubits when needed, and (ii) choose angles so that selected vertices appear random.

Randomly interleaving which rounds are tests and which vertices host traps gives blindness over trap placement. The acceptance is decided when we know how

many test rounds are correct, giving an idea of how many computation rounds are unaffected.

Choosing this trappification scheme, there is no space overhead per round. Computation and tests live in different rounds. Every qubit of a computation round serves the target MBQC; every qubit of a test round is either a single trap-qubit or a dummy. Hence a single-round costs exactly what the unprotected UBQC would.

Embedding

Here the embedding module is nothing but repetition:

- Choose d indices for computation rounds $C \subset [n]$, run the target MBQC there.
- Run $n - d$ test rounds drawn from the trappified scheme.
- Decode by majority vote on the d outcomes.

The amplification relies on the embedding of the computation result into a classical repetition code. For a BQP instance with error c—otherwise said with a gap bigger than c—the harmless deviations are those that flip less than a fraction $\frac{2c-1}{2c-2}$ of the computation rounds. For in this case, it is impossible to flip the result of the majority vote. Therefore, these should be included into the insensitive set. To this end, we adjust the acceptance rule to accept as many as possible of these harmless deviations. Given that a computation with a single deviated qubit might yield the wrong result, the best we can do is to have at most $\frac{2c-1}{2c-2} \times d$ computation rounds affected by a deviation. This in turn can be probed by the test rounds thanks to the randomization of their locations and the overall blindness. One only needs to keep in mind that test rounds do not detect deviations with probability 1. For universal graphs like the Brickwork state, this is equal to 0.5 as the chance of a given qubit to be a trap in a test round is $\frac{1}{2}$.[7] Putting everything together, if we set the acceptance function to output 0 as long as the fraction of failed test rounds is below and bounded away from $\frac{1}{2} \times \frac{2c-1}{2c-2}$ by a constant then the accepted deviations are guaranteed to be harmless.

As a consequence, the protocol remains correct and accepts with high probability even when the prover is noisy, provided the global fraction of failed rounds stays below and bounded away by a constant from the threshold of $\frac{1}{2} \times \frac{2c-1}{2c-2}$.

[7] This allows to have only two types of test rounds while ensuring a uniform coverage of all qubits. In practice, this means that a signle qubit deviation will be detected with a probability independent its location. From a security point of view, this seems optimal, yet with respect to noise robustness, the question is largely unexplored.

5.3 *Zero Space-Overhead Robust Verification*

Using the framework introduced earlier, it is rather straightforward to prove the security of Protocol 6.

Theorem 6. Security of Protocol 6

> *Let G be a k-colorable graph, and C a computation defined as an MBQC pattern on G. For $n = d + t$ such that d/n and t/n are fixed in $(0, 1)$ and w such that w/t is fixed in $(0, \frac{1}{k} \cdot \frac{2c-1}{2c-2})$, where c is the inherent error probability of the* BQP *computation, Protocol 6 with d computation rounds, t test rounds, and a maximum number of tolerated failed test rounds of w is ϵ-composably-secure with ϵ exponentially small in n.*

The fact that we have such sharply distinct behavior between aborting with probability one and accepting with probability one in case of low noise signals the interest of these verification techniques for probing noise strength. This is an avenue that is currently pursued for setting benchmarking frameworks of quantum computers and networks that was indeed pioneered in the Quantum Internet Alliance project cite:AFCD23requirements. This possibility offers an unexpected leverage to encourage hardware makers to integrate the necessary features for implemeting verification into their devices as it not only serves a utility-regime protocol—verification—but also fulfills a more immediate need of providing certifiable benchmarks for their early machines.

This clearly demonstrates that the absence of space-overhead is not detrimental to efficiency as the security error remains negligible in the total number of round n.

The robustness can be obtained likewise. One can even obtain that setting the threshold too low would yield to overwhelmingly aborting the computation.

Theorem 7.

> *As before, c denotes the inherent error probability for the* BQP *computation instances running as MBQC patterns on a k-colorable graph G. Assume a Markovian round-dependent model for the noise on Verifier and Prover devices and let $p_{\min} \leq p_{\max} < \frac{1}{k} \cdot \frac{2p-1}{2p-2}$ be respectively a lower and an upper-bound on the probability that at least one of the trap-qubit measurement outcomes in a single test round is incorrect. If $w/t > p_{\max}$, Protocol 6 is ϵ_{cor}-correct with exponentially low ϵ_{cor}. On the other hand, if $w/t < p_{\min}$, then the probability that Protocol 6 terminates without aborting is exponentially low.*

5.4 Takeaways

We have shown that using UBQC wrapped in a schedule of computation and test rounds together with a majority vote enable a verification protocol that is both prover-light and noise robust.

In summary:

- Every qubit in a computation round serves the target task; traps live in separate test rounds.
- As outputs are classical, we boost detection via majority vote, not heavy fault-tolerant codes.
- The threshold on failed tests trades amplification strength against robustness to honest noise.
- RSP is the plain UBQC sender-prepares primitive; trappification are standalone trap rounds; embedding is repetition and majority decoding.
- Local guarantees—detection, insensitivity, correctness, are lifted to global AC security, while honest-but-noisy provers still pass.

This sets the stage for the next chapters: slimming the verifier further, adding multi-party features, and eventually handling fully fault-tolerant workloads.

6 Dummyless VBQC

The goal of this chapter is to show, by direct instantiation of the modular framework, that verifiable blind delegation does not actually need dummies~(see Section 2.5.2) when the target computation has classical input and classical output. In VBQC, dummies serve to disconnect parts of the graph and insert traps. Here we replace them entirely by (i) an intrinsic symmetry of classical-Input/Output MBQC, which makes one specific non-trivial Pauli error harmless, and (ii) stabilizer traps built only from $\mathsf{I}, \mathsf{X}, \mathsf{Y}$ operators. The outcome is a protocol that:

- Keeps blindness.
- Detects every harmful deviation with negligible failure probability.
- Imposes no extra space overhead on the prover: every qubit in a computation round can be used for the computation as traps live in separate rounds.
- Uses the same RSP instantiation as robust VBQC (Protocol 6) but with states restricted to XY plane.

Beyond immediate practicality (simpler sources, simpler calibration), the exercise is conceptually useful: it stresses how the framework let us push invariances into the insensitive set and trim the trappification module accordingly, while preserving full composable security. More importantly, the obtained protocol will be used for all subsequent improvements.

6.1 Protocol

The dummyless protocol is a direct instantiation of VBQC (Protocol 3) with specific choices of RSP, Trappification and Embedding. Indeed, the next section will first instantiate the Trappification as it will unlock the ability to have an RSP with only states in the equatorial plane. The embedding will be the repetition as we will restrict ourselves to BQP computations.

6.2 Module Definitions

Dummyless Trappification

A Natural Invariance of MBQC with Classical Input and Output

In MBQC, computation qubits, i.e. $v \in O^c$, are measured in the $\left|\pm_{\phi'(v)}\right\rangle$ basis, with $\phi'(v) \in \Theta = \left\{\frac{k\pi}{4}\right\}_{k\in\{0,\dots,7\}}$ fixed by the pattern. Because a projective measurement only depends on its projectors, any unitary that leaves those projectors invariant will not change the measurement statistics. When measuring along the $\phi'(v)$-axis in the XY plane, any rotation around that axis let the outcome statistics unchanged. This invariance is routinely used in UBQC security proofs to twirl the prover's attack.

The same observation extends beyond unitaries. If we allow any local invertible map that reflects the Bloch sphere with respect to the XY plane, the projectors onto $\left|\pm_{\phi'(v)}\right\rangle$ are still left unchanged. Hence, measurement probabilities are unaffected. When *all* qubits are measured in the XY plane—corresponding to classical-I/O—this invariance propagates to the final classical result.

To this end, we introduce a linear map F_A that flips the sign of every Z term on a chosen subset $A \subset V$ in the Pauli expansion of a state ρ on V. More formally, for

$$\rho = \sum_{\mathsf{P}\in\{\mathsf{I},\mathsf{X},\mathsf{Y},\mathsf{Z}\}^{\otimes n}} \alpha_{\mathsf{P}}\mathsf{P}, \text{ then} \tag{6.1}$$

$$F_A(\rho) = \sum_{\mathsf{P}\in\{\mathsf{I},\mathsf{X},\mathsf{Y},\mathsf{Z}\}^{\otimes n}} (-1)^{\mathrm{zwt}_A(\mathsf{P})}\alpha_{\mathsf{P}}\mathsf{P}, \tag{6.2}$$

where $\mathrm{zwt}_A(\mathsf{P}) = |\{v \in A | \mathsf{P}_v = \mathsf{Z}\}|$ counts the number of vertices in A on which P equals the Pauli Z. Then we have:

Lemma 1.

Let $G = (V, E)$ and an MBQC pattern on G measuring all qubits in V. Then for any $A \subset G$, applying F_A right before the XY plane measurements leaves the output distribution unchanged.

The map F_A is not generally a unitary, but it is still possible to find a unitary transformation that has the same effect as F_A on $|G\rangle\langle G|$. As a result, $F_A\left[|G\rangle\langle G|\right]$ is a physical state:

Lemma 2.

For any graph $G = (V, E)$ it holds that $F_A\left[|G\rangle\langle G|\right] = \mathsf{U}\,|G\rangle\langle G|\,\mathsf{U}^\dagger$, where

$$\mathsf{U} = \prod_{\substack{v\in V,\\ \deg v \equiv 1 \,(\mathrm{mod}\ 2)}} \mathsf{Z}_v \tag{6.3}$$

describes the application of Z operators to all odd-degree vertices of G.

Putting both lemmas together yields the specific harmless deviation that will matter later:

Theorem 8.

Let $G = (V, E)$ be a graph and E^* be the unitary operation given by

$$\mathsf{E}^* = \prod_{\substack{v\in V,\\ \deg v \equiv 1 \,(\mathrm{mod}\ 2)}} \mathsf{Z}_v, \tag{6.4}$$

describing the application of Z operators to all odd-degree vertices of G. For MBQC on G with classical input and output, the application of E^* before the measurements has no effect on the results of the computation.

In short, for any classical-I/O MBQC there exists a non-trivial, generally non-stabilizer Pauli deviation—the product of Z's on odd-degree vertices—that is undetectable by our forthcoming dummyless traps, yet harmless for correctness. We should exclude it explicitly from the set of errors we try to catch as we pay no security price in doing so.

Trappification Scheme

We now exploit this newly found invariance. Because we restrict ourselves to preparations in the XY plane and MBQC authorizes only XY measurements, every trap we insert must use only those resources. Concretely, we want stabilizers of the graph state that never ask us to prepare or measure a Z eigenstate. That pushes us to look for generators of the stabilizer group of $|G\rangle$ made only of $\mathsf{I}, \mathsf{X}, \mathsf{Y}$. This is because preparing an eigenstate of a generator containing a Z at a given location is done by preparing a $|0\rangle$ state for this location.

Intuitively, we need to (i) build as many independent, deterministic X, Y stabilizer tests as the graph allows; (ii) randomize over them; (iii) ignore the one residual deviation that we proved harmless. We will show that this is indeed enough to detect all harmful deviations.

The following lemma formalizes the first step.

Lemma 3.

> *Let $G = (V, E)$, the graph state $|G\rangle$ and its stabilizer group S. There exists $|V| - 1$ generators of S that are tensor products of I, X and Y only. Denote by R the subgroup generated by those $|V| - 1$ elements.*

We can now define our trappified scheme $\boldsymbol{P}$ by sampling uniformly at random one of these $|V| - 1$ generators and running the corresponding stabilizer test traps (Section 4.2).

Definition 12.

> A dummyless trappified scheme $\boldsymbol{P}$ with parameters N, d for a BQP computation on graph G is obtained by considering N rounds that will be delegated using UBQC on graph G. d rounds will be sampled uniformly at random to host the delegated computation, and for each of the remaining $N - d$ ones, a trap will be sampled at random by choosing one of the $|V| - 1$ stabilizer of G that contain only $\mathsf{I}, \mathsf{X}, \mathsf{Y}$ operators.

Because of the previous lemma, one stabilizer generator of $|G\rangle$ is outside R. Call it S_0. Using the analysis of the previous section, it is in fact easy to discover. It consists of a tensor product of Z's at the odd-degree vertices. As a consequence, all the Pauli operators that commute with R and are not in R—these are non-trivial undetectable errors—are necessarily of the form $\mathsf{S}_0\mathsf{R}$. Because they commute with R they cannot be detected (see Fig. 6.1). Fortunately, Theorem 8 shows that S_0 is indeed harmless for the computation and so are all the elements in $\mathsf{S}_0\mathsf{R}$.

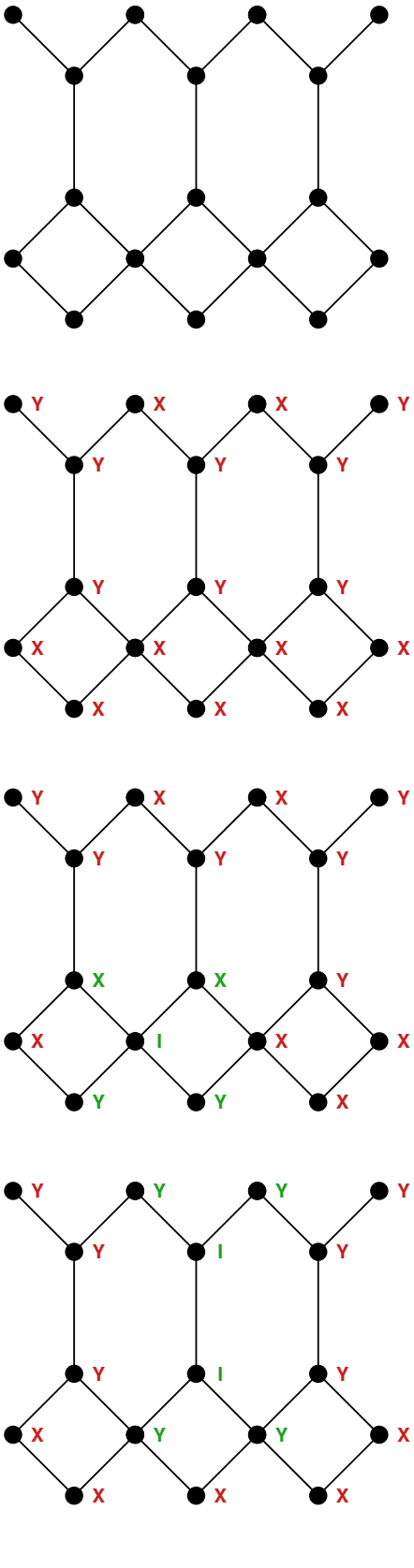


Figure 6.1: Traps based on "poking holes" on even degree vertices (up), traps based on chains linking two odd degree vertices (middle), the uncaught deviation that has no effect on BQP computations (down).

Remote State Preparation

Nothing fancy is needed here: we reuse the RSP construction of Protocol 7. The only difference for the dummyless setting is that now the verifier never has to send $|0\rangle, |1\rangle$ states. Indeed, $|+_\theta\rangle$, $\theta \in \Theta$ suffice.

Protocol 8. Single-Plane RSP

> **Public Information:** Θ
> **Alice's Input:** $\theta \in \Theta$
> **Preparation of the state:**
> - Alice prepares $|+_\theta\rangle$ in a quantum register.
>
> **Transmission of the state:**
> - Alice sends the quantum register prepared in $|+_\theta\rangle$ to Bob.

As in the robust case, the resource is secure-by-design: Bob learns only what the quantum state itself reveals, and any later deviation cannot depend on θ.

Embedding

Just as in robust VBQC (Protocol 6), amplification is obtained by repetition and majority vote: classical-I/O lets us avoid using fault-tolerant quantum computation procedures. The embedding algorithm here is also the same: run the same computation round d times; interleave with t trap rounds sampled from $\boldsymbol{P}$; post-process by majority vote over the d computation rounds.

The properness of the embedding algorithm—no information flow from computation to traps—is ensured by keeping rounds disjoint.

6.3 *Verification without dummies*

Using the previous definitions we can now apply Theorem 5 using the trappified scheme $\boldsymbol{P}$ defined earlier (Definition 12) that:

1. Detects the set $\mathcal{E}$ of harmful Z-Pauli errors.
2. Correctly evaluates the target computation in the presence of any other Z-Pauli error in at least $\mathcal{P}_V^{\mathsf{Z}} \setminus \mathcal{E}$. Theorem 8, then shows that there is a specific error E^* which never affects the output distribution of the target computation and thus does not need to be detected.

This results in:

Theorem 9.

Let $G = (V, E)$ be a graph, and $\boldsymbol{P}$ the trappified scheme on G defined by sampling at random from a generating set of R *containing only stabilizers with no* Z*'s. Then, $\boldsymbol{P}$ constructs the SDQC Resource 2 for* BQP *computations that can be embedded on the graph G with negligible correctness error and security error.*

6.4 Takeaways

We have shown that dummies are not intrinsically required for verifying quantum computations. This rests on:

- Building traps entirely in the XY plane using stabilizer generators without Z's.
- Recognizing and factoring out a single unavoidable, but harmless, deviation.
- Reuse the repetition-based amplification and acceptance tuning of robust VBQC.

Practically, the verifier no longer needs to prepare computational basis states, simplifying hardware; the prover still sees the same per-round resource state as in VBQC; and composability is preserved by working inside the framework (Chapter 4).

Part III
Reducing Requirements for the Verifier

This Part explores two ways in which the requirements for verification can be reduced on the Verifier's side. In both cases the goal is to equip verifiers with good quality, robust and cheap devices. The first one deals with verifiers without the ability to prepare quantum states and relying on the prover to do that for them. The second is about verifiers using weak coherent pulses—obtained by attenuating a laser—instead of single-photon sources. The difficulty with those changes is that the devices now provide less advanced or even imperfect functionalities. As a consequence, this needs to be compensated by more complex protocols, whose security is harder to obtain because of the increased amount of information exchanged and the number of interactions with the prover.

7 Trusted Rotations

The objective of this chapter is to lighten the verifier: no single-photon source, no state preparation. Instead, we assume only that the verifier can apply trusted single-qubit Z-rotations and random X flips to whatever qubit the prover hands over. Because Remote State Preparation (RSP) is a composable module, it suffices to replace it locally by an equivalent primitive. We show that a Remote Rotation with Dephasing resource achieves this for BQP computations, provided the prover is trusted to send qubits.

7.1 Motivation

VBQC-like protocols assume the verifier can prepare $|+_\theta\rangle$ states, and sometimes also $|0\rangle, |1\rangle$ states. This implies some relatively heavy set-up for the verifier: good near-deterministic photon sources require cryogenics and careful calibration and maintenance. If the prover could prepare the raw qubits and the verifier only perform additional trusted rotations, life would be easier.

This question was first raised as open in [DKL12]. In [MKA+22] it was shown to provide blindness. We exhibit below a solution that is statistically secure but has one caveat. The source is dimension-bounded: the verifier is guaranteed to receive *bona fide* qubits.

Other approaches provide verification with light verifiers. [FKD18] and [FKD19] remove the need for trusted preparation by considering some physically motivated noise—thus do not provide verification in the fully malicious scenario. This is in addition to fully classical verifier verification that holds under computational assumptions [Mah18], but which incurs a large overhead for the prover.

7.2 Security Failures Without Trusted Preparations

Here we want to dispel a confusion that we have seen repeatedly in published papers: Blindness alone does not stop an adversary from mounting selective attacks on dummy qubits in VBQC type protocols. Consider the following idealized resource where Alice instructs a unitary on a state held by Bob:

Resource 4. Remote Unitary (RU)

> **Alice's Input:** A unitary U chosen from $\{\mathsf{H}, \mathsf{XH}, \mathsf{Z}(\theta) \text{ for } \theta \in \Theta\}$.
> **Bob's Input:** A state ρ
> **Computation by the Resource:** The Resource ouputs $\mathsf{U}[\rho]$ at Bob's interface.

A trivial implementation consists for Bob to send a state ρ to Alice; Alice applies U and sends it back. The problem with this resource is that it does not allow performing verified quantum computation—irrespective of its implementation. Indeed, it lets the prover selectively attack the dummy qubits, even though it does not know which of the qubits are dummies.

For instance, instantiate the RSP with RU where the prover's input is set to $|+\rangle$. Let the verifier decide to prepare a dummy. The verifier simply needs to perform H on the provided state and send it back. If it wants to prepare a $|+_\theta\rangle$ state, it simply performs a $\mathsf{Z}(\theta)$ rotation. The problem is that the prover could decide to cheat by sending $|-\rangle$ instead of $|+\rangle$, and apply Z whenever it receives the qubit back from the verifier. A quick calculation shows that $|+_\theta\rangle$ states are unaffected, but dummies

get an X deviation (See Fig. 7.1). This provides a way to craft a concrete selective attack on the dotted-triple-graph variant of VBQC [KW17] (Protocol 3).

This is a manifestation of a deeper phenomenon: deviations or noise can pick-up a dependency on the secret parameters used to parameterize the preparation circuit and break the security proofs of the protocols. This will become a crucial observation and a difficulty in designing protocols for fault-tolerant quantum computations, as honest noise occuring during the preparation of states by the verifier can pick-up dependency upon secret parameters which voids the security of naive protocols.

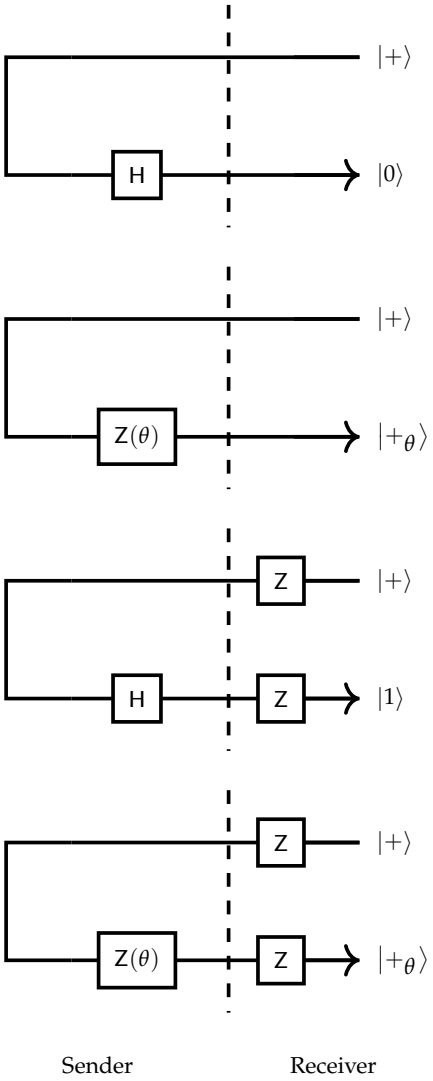


Figure 7.1: Concrete attack on dummy qubits obtained by having the receiver provide $|-\rangle$ states instead of a $|+\rangle$ states.

7.3 Recovered Security with Remote State Rotation

To neutralize the above attack and remove any potential ones due to secret-dependency picked by an incorrect preparation, we switch to the dummyless setting. The sender in the RSP only needs to ensure $|+_\theta\rangle$ states are available at the receiver's interface once its actions are done. The difficulty is that we want to allow whatever single-qubit state the receiver[8] is providing to the sender, both parties still end up with a secure, composable RSP.

The selective-attack from the previous subsection—hidden Z before and after—no longer works: Z commutes with $Z(\theta)$. But a subtler problem remains. If the Prover sends a state that is not in the XY plane, say $Y(\alpha)|+\rangle$ for $\alpha \in [0, \pi/2]$, then generally $Z(\theta)Y(\alpha)|+\rangle \neq Y(\alpha)Z(\theta)|+\rangle$, so the deviation cannot be pushed "after" the resource. In composable terms, the prover's pre-rotation now depends on the verifier's secret θ once the operations are reordered. This then breaks the security of the implementation of the RSP resource.

The cure is to force the incoming state into the XY plane before applying $Z(\theta)$. A uniformly random X flip does exactly that: it decoheres in the X basis, wiping any Z-component that could pick up θ. This leads to the following resource where Alice plays the sender and Bob the receiver:

Resource 5. Remote Rotation with Dephasing (RR_D)

Inputs:
- Alice inputs an angle $\theta \in \Theta$.
- Bob inputs a single-qubit in state ρ.

Computation by the Resource:
1. The Resource samples a bit $b \leftarrow\$ \{0,1\}$ uniformly at random.
2. The Resource outputs a qubit in state $Z(\theta)X^b[\rho]$ to Bob.

Constructing the 8-state XY plane RSP from RR_D is done easily (see Fig. 7.2

Protocol 9. Single Plane RSP from RR_D

Input: Alice inputs an angle $\theta \in \Theta$.
Protocol:
1. Alice and Bob call the RR_D Resource 5:
   - Alice inputs the angle θ.
   - Bob inputs a qubit in the state $|+\rangle$.
   - The Resource returns a single-qubit to Bob who sets it as its output.

The following theorem ensures that the construction is secure:

Theorem 10. Security of Protocol Protocol 9

Protocol 9 perfectly constructs Resource 3 where the states alphabet $\Sigma = \Theta$ from Resource 5.

[8] We have kept the terminology of sender, receiver from the RSP resource, yet in our setting the state that is available to the sender is prepared by the receiver. This could be because the receiver initially sends it over a quantum channel, or more likely this corresponds to a situation where the source that is used to create the initial states upon which the sender will apply U is potentially malicious.

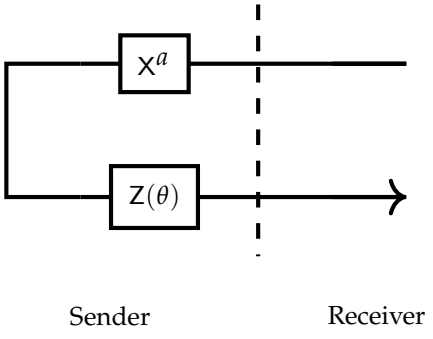


Figure 7.2: Single Plane RSP from RR_D protocol.

7.4 Verification with Remote State Rotation

Thanks to composability, nothing dramatic happens at the protocol level: in dummyless VBQC—which follows Protocol 5—replace each call to the single plane RSP by Protocol 9, i.e. use RR_D under the hood. This constructs an SDQC resource with

the same security parameters. The construction error for RSP is zero when using RR_{D}, so the global bound is untouched.

There is, however, one caveat worth flagging explicitly. In the informal discussion above we implicitly assumed the prover still hands over a single-qubit. If instead higher-dimensional systems can slip through—e.g., two photons when one is expected—the random-X dephasing no longer guarantees that the prover's deviation is θ-independent. A concrete threat in photonic implementations is the photon-splitting attack: send two photons, let both accumulate the secret $\mathsf{Z}(\theta)$, then split and measure to glean θ. Curbing this requires an extra assumption—trusted dimension—or an explicit countermeasure—which could be provided by the WCP techniques discussed later. The moral is rotations-only RSP solves the preparation problem, not the dimension problem.

7.5 *Takeaways*

We have shown that stripping the verifier down to trusted $\mathsf{Z}(\theta)$-rotations and random \$X's is enough for verifiable delegation—provided the prover really sends a qubit.

Two broad lessons emerge. Firstly, a single phase modulator and a fast bit flip suffice to verify computations. Blind-only schemes such as [MKA+22] can be upgraded to full verifiability by a drop-in replacement of their RSP and appropriate trappification changes. Secondly, negligible security error survives. Because RR_{D} perfectly constructs RSP, any VBQC variant using RSP inherits its proof unchanged. Yet, this also emphasizes that blindness is not a force field. Even if the prover cannot locate dummies, it can still craft attacks that selectively hurt specific positions—or their complements. Several papers overlook this subtlety, undermining their claims.

8 Weak Coherent Pulses

Our goal in this chapter is to replace the single-photon Remote State Preparation module by one implementable with weak coherent pulses (WCP). Concretely:

- We keep the dummyless VBQC; only the RSP module is replaced.
- We model multi-photon leakage and take into account the loss acknowledgment by the prover as the attack lever, then build a resource that still delivers at least one secret qubit per batch.
- We recover verifiability by combining that batch resource with the phase-accumulation gadget of [DKL12].
- We improve over [DKL12] by providing verifiability and practicality: the number of pulses scales like $1/\eta^2$—where η is the channel transmittance—instead of $1/\eta^4$ in the original scheme.

In short, this chapter shows how to (i) verify while tolerating realistic laser sources, while (ii) maintaining acceptable performance in spite of the much larger attack surface of the RSP module.

8.1 Motivation

The previous chapter argued that single-photon sources are a costly bottleneck on the verifier's side. Weak coherent pulses (WCPs) offer a different incentive: they propagate farther through fiber—thanks to higher usable launch power, mature telecom hardware—and they are already ubiquitous in QKD deployments. If we could base an RSP construction on WCPs, long-distance verified delegation could possibly ride on existing QKD infrastructure with minimal extra optics.

Two issues must be faced head-on:

- *Multi-photon leakage and acknowledgements.* A WCP occasionally contains 2+ photons. A malicious prover can keep one, measure the rest, and learn the secret angle θ. Because the channel is lossy, the prover must be allowed to confirm reception; he can craft a successful attack by simply declaring all 0- and 1-photon pulses lost and accept only exploitable ones from his malicious perspective.
- *Incomplete or non-composable security in earlier work.* The original WCP proposal of [DKL12] is phrased in a composable framework, but it only guarantees blindness as it couldn't produce dummies and hence couldn't verify. Later decoy-state based refinements [JWH+19] improve efficiency but abandon a full AC treatment. Additionally, we show a concrete flaw that undermines their claim to security for VBQC—and even affects the standard QKD decoy analysis.

Our approach is to stay strictly within Abstract Cryptography. We model a realistic WCP device as a resource, prove how to construct from it a batch containing at least one qubit prepared in a genuine $|+_\theta\rangle$ state, and plug that *batchRSP* resource into the dummyless VBQC (Protocol 5 with dummyless tests). The reward is composable security against fully malicious provers; and by using multiple intensities, a scaling that improves from $1/\eta^4$ to $1/\eta^2$ for a fiber with transmittance η.

8.2 Security Failures Due to Multi-Photon Pulses

Remote State Preparation's security asks that the receiver[9] learns nothing beyond

[9] Here again we switch to the RSP terminology of sender / receiver as it could be used outside the verification setting of verifier / prover.

the quantum state it receives, and that any imperfection can be pushed into a deviation after delivery. Weak coherent pulses (WCP) depart from this specification in a decisive way.

An attenuated laser does not emit single photons on command, it emits pulses whose photon number is Poisson distributed with mean μ. Vacuum shots are harmless, single-photon shots are fine, but as soon as two—or more—photons slip through, a malicious receiver can split the pulse. One (or more) photon is measured to glean information about the secret angle θ; at least one is kept pristine to satisfy later checks. The deviation has now acquired a dependence on the secret. Composable security forbids exactly this: whatever imperfection over the ideal resource must map to a deviation independent of θ and applied after the perfect state is delivered.

Indeed, the protocol itself hands the adversary the steering wheel. Because most pulses are empty and the channel is lossy, the sender must let the receiver acknowledge which pulses arrived. That acknowledgment is a perfect filter: discard every 0- or 1-photon event, keep only the multi-photon ones you can exploit. Blindness does not save us here; the filter acts on photon number, not on the intended preparation.

The verdict is immediate. An RSP constructed naively from WCP source is not secure. The leakage of θ on multi-photon rounds cannot be swept under the rug and treated as a post-resource deviation. Unless we alter the construction by constraining the acceptance rule so that any profitable photon-number filtering becomes statistically visible, verification fails before trappification or embedding even enter the stage.

8.3 Recovered Security and Performance

The cure comes in two doses. First, we pin down exactly what a weak coherent pulse source gives an honest or a cheating receiver. Second, we add classical statistics and a tiny quantum gadget to make any profitable multi-photon filtering show up in those statistics. We then provide two statistical tests. The one of [DKL12] which gives an asymptotic $1/\eta^4$ scaling and one inspired by the decoy state method that scales as $1/\eta^2$.

Modeling the source: WCPGenerator

We wrap the physical laser and lossy fibre in a resource that is deliberately pessimistic for the dishonest case. If a proof survives this model, it survives the lab. We call this resource WCPGenerator.

When the receiver is honest, it receives n photons drawn at random from Poisson$(\mu\eta)$—μ is the power of the laser and η is the channel transmittance—prepared as $(\mathsf{U}[\rho])^{\otimes n}$. When it cheats, it gets the lossless distribution Poisson(μ), and if $n > 1$ it is even handed the classical description of U. For Alice playing the sender and Bob the receiver this gives the following resource:[10]

[10] Here we emphasize again that Resource 6 is not a physically realistic resource, but rather a model that is pessimistic enough so that reasonable physical resources are more powerful than this one. As a consequence, if we prove security with this resource, we will have security with a more powerful resource.

Resource 6. WCPGenerator

Public information: set of unitaries $\mathcal{U}$; classical description of a quantum state ρ, power of the laser μ, transmittance $\eta \in [0,1]$.
Alices's Input: $\mathsf{U} \in \mathcal{U}$ and $\mu \in \mathbb{R}_0^+$.
Bob's Input: $c \in \{0,1\}$, set to 0 if honest.
Computation by the Resource

- $c = 0$, it samples $n \leftarrow\!\!\$$ Poisson$(\mu\eta)$ and sends the state $\mathsf{U}[\rho]^{\otimes n}$ to Bob.
- $c = 1$, it samples $n \leftarrow\!\!\$$ Poisson(μ). If $n \leq 1$, it sends the state $\mathsf{U}[\rho]^{\otimes n}$ to Bob. If $n > 1$, it sends n and the classical description of U to Bob.

Phase–accumulation gadget (from [DKL12])

A gadget does the heavy lifting for giving the sender some handle over the receiver. Label qubits $0, \ldots, N$ and prepare them in the XY plane. Apply CNOT's with control on qubit 0 and target on qubit i for all $i \geq 1$. Then measure qubits 1 to N in the

computational basis. Up to the known outcomes, the phase angles add on qubit 0. One unknown angle therefore serves as a one-time pad for the output phase.

From a security perspective, if among N pulses at least one is truly single-photon, the gadget can channel that secrecy into a single clean qubit (See Fig. 8.1).

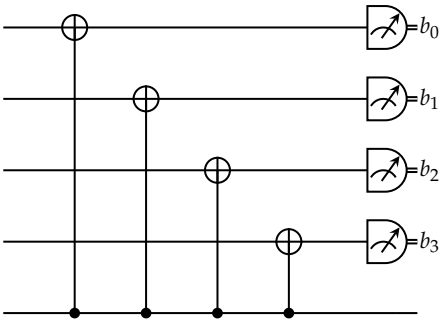


Figure 8.1: This gadget produces the quantum state $|+_\theta\rangle$ where $\theta = \theta_0 + \sum_{i=1}^{n} (-1)^{b_i}\theta_i$.

First recovery: single-intensity defense (from [DKL12])

The insight of [DKL12] is twofold. First, given the phase–accumulation gadget, one single genuine single-photon is enough to hide the overall phase. This is because of the additive nature of the overall phase that is one-time-padded by the unknown phase of the single-photon pulse. Hence the optimal cheating move for the receiver is obvious: make sure the sender never gets a lone-photon pulse. Declare every 0- and 1-photon pulse "lost", keep only 2-or-more-photon pulses. The gadget then has no unknown angle left to channel.

Second, the gadget itself gives the sender a handle to fight back—by simple counting. In the honest case, the number of pulses the receiver cannot acknowledge is governed by the WCP intensity and the loss: essentially the fraction of 0-photon events after the channel. In the dishonest strategy above, every 0- and 1-photon emissions must be discarded to prevent secrecy from sneaking through. By tuning the WCP intensity appropriately, the first one can be made smaller than the second by a constant margin allowing efficient statistical testing.

This translates into the following actions of the sender:

1. He chooses a mean photon number μ that fulfills the above condition.
2. He checks the receiver's "not-received" reports against the honest prediction. Too many discards would yield an abort; the multi-photon filtering cannot be hidden.

This restores the secure RSP construction with a single intensity and a single threshold test. But there are two shortcomings in the original proposal–besides the absence of verifiability of the whole:

- The resource cost explodes as $1/\eta^4$ when the fiber transmittance η becomes small making it inefficient for long distance.
- Photon-number–resolving detection were assumed on the receiver even when he is honest; strictly speaking, that is unnecessary once we look closer at the proof. These are the issues the next subsection fixes by moving to two intensities and constructing *batchRSP*.

Recovered Security and Performance

The solution proposed below is described in [GLMO24] and later refined into [GLMO25]. It is based on the previous approaches of [DKL12] for the gadget and general approach toward the proof and on [JWH+19] for the motivation behind using decoy states. Instead of going for the full RSP, we focus instead on the creation of a batch of size K where at least one pulse has a single photon. Indeed, this is enough to construct RSP using the previously defined gadget.

Resource 7. BatchRSP

> **Public Information:** $K \in \mathbb{N}^+$; set of unitaries $\mathcal{U}$; classical description of a quantum state ρ.
> **Alice's Input:** K unitaries $\mathsf{U}_1, \ldots, \mathsf{U}_K \in \mathcal{U}$.
> **Bob's Input:** $c \in \{0, 1\}$, set to 0 if honest.
> **Computation by the Resource**
> - $c = 0$, it sends the states $\mathsf{U}_1[\rho], \ldots, \mathsf{U}_K[\rho]$ to Bob.
> - $c = 1$
> - It samples $k \leftarrow\!\!\$\ \{1, \ldots, K\}$ uniformly at random.
> - It sends k, the classical descriptions of $\{\mathsf{U}_i\}_{i \neq k}$, and the quantum state $\mathsf{U}_k[\rho]$ to Bob.

To ensure that BatchRSP has at least one pulse with a single photon, we construct a statistical test similar in spirit to the one above. However, this time we will use 2 intensities.

Informally, the sender will generate pulses deciding their intensity randomly. He will keep track of the intensity of the acknowledged pulses. Then it will compute a quantity using the number of pulses acknowledged for each intensity.

More concretely, let ν and ν' be the two intensities and P and P' the corresponding number of acknowledged pulses. Define $a = e^{-\nu}, b = \nu e^{-\nu}, c = \nu^2 e^{-\nu}/2$ and similarly for a', b', c' where ν is replaced by ν'. These are the expected fractions of 0, 1 and 2 photons pulses in a WCP of intensity ν and ν'. Next we define two numbers that the sender can compute on its own given the intensities of the laser it uses, the expected value of the transmittance η and the reported acknowledged pulses:

$$T = \frac{c'P - cP'}{bc' - b'c}. \tag{8.1}$$

$$t = \frac{c'(1 - e^{-\eta\nu}) - c(1 - e^{-\eta\nu'})}{bc' - b'c} \frac{N}{2}. \tag{8.2}$$

For a total number of N pulses—half for each intensity—and with I the set of acknoledged pulse, the sender accepts or aborts using the following algorithm:

Algorithm 1. Algorithm $\mathcal{B}$

Input: I
Estimation:
- Fix $\Delta_0 > 0$.
- Compute t and T.

If $T \geq t - \Delta_0 \frac{N}{2}$, return Accept.
Otherwise, return Abort.

The value of Δ_0 is a security margin for the statistical test that essentially gives a higher confidence in the decision when Δ_0 is bigger for a fixed N.

This can be summarized in a full protocol:

Protocol 10. Multi-Intensity Weak Coherent Pulse Method

Public Information: $N, K \in \mathbb{N}^+$, where $K \leq N$; group of unitaries $\mathcal{U}$; classical description of a quantum state ρ ; transmittance $\eta \in [0,1]$; pulse intensities $\nu, \nu' \in \mathbb{R}_0^+$; efficient, classical estimation algorithm $\mathcal{B}$.
Alices's Input: K unitaries $\mathsf{U}_1, \ldots, \mathsf{U}_K \in \mathcal{U}$.
Alice - Sending WCP
- Sample a random permutation $\pi \leftarrow\!\!\$\ \mathsf{S}_N$.
- Let $(\nu_1, \ldots, \nu_N) \leftarrow (\nu_{\pi(1)}, \ldots, \nu_{\pi(N/2)}, \nu'_{\pi(N/2+1)}, \ldots, \nu'_{\pi(N)})$.
- $i \in [1, N]$
  - Sample $\mathsf{U}'_i \leftarrow\!\!\$\ \mathcal{U}$ from the Haar measure on $\mathcal{U}$.
  - Call WCPGenerator with $\mathsf{U} = \mathsf{U}'_i$, $\mu = \nu_i$, the classical description of the quantum state ρ, and transmittance η.

Bob - Acknowledging WCP Reception
- Let $\tilde{I}$ be the set of rounds whose pulses contain at least one photon.
- if $|\tilde{I}| < K$ Send Abort to the Sender and stop.
- Let $I' \leftarrow$ random subset of $\tilde{I}$ of size K.
- For $i \in I'$, store one round-i photon in quantum memory.
- Send I' to the Sender.

Alice - Estimation
- Undo the permutation, i.e. let $I \leftarrow \{\pi^{-1}(i) | i \in I'\}$.
- Run estimation algorithm $\mathcal{B}$ on input I.
  - If $\mathcal{B}$ returns Abort, send Abort to Bob and stop.
- Relabel the set of kept states in I' from 1 to K. Sample a random permutation $\sigma \leftarrow\!\!\$\ \mathsf{S}_K$.
- For $j \in [1, K]$, let $\tilde{\mathsf{U}}_j = \mathsf{U}_{\sigma(j)} \mathsf{U}'^{\dagger}_j$.
- Send the corrections σ, $(\tilde{\mathsf{U}}_j)_{j \in \tilde{I}}$ to Bob.

Bob - Corrections
- For $j \in [1, K]$, apply the unitary $\tilde{\mathsf{U}}_j$ to the j-th kept state.
- Permute all kept K quantum states using σ, and set the result as the output.

The following theorem then gives the security error of the construction when compared to the ideal BatchRSP (Resource 7):

Theorem 11. Correctness and Security Errors

Given $\Delta_0 > 0$ in Algorithm $\mathcal{B}$, $\delta > 0$ such that $K = ((2 - e^{-\eta\nu} - e^{-\eta\nu'})/2 - \delta)N$ and $C = \max(c, c')$, the correctness error ε_{corr} satisfies

$$\varepsilon_{corr} \leq \exp\left(-\delta^2 N\right) + \exp\left(-\frac{\Delta_0^2(bc' - b'c)^2}{4C^2}N\right) \tag{8.3}$$

while the security error is negligible in N whenever there are additional constants $\Delta_0', \Delta_0'' > 0$ such that

$$\begin{aligned}\Delta_0 + \Delta_0' + \frac{c'}{bc' - b'c}\Delta_0'' \\ = \frac{c'(1 - e^{-\eta\nu}) - c(1 - e^{-\eta\nu'}) - c'(1 - a - b - c)}{bc' - b'c}.\end{aligned} \tag{8.4}$$

In addition, in the high-loss limit $\eta \to 0$, the number of pulses needed is given by the following theorem:

Theorem 12. Scaling for $\eta \to 0$

For $\nu = \alpha\nu'$ with $0 < \alpha < 1$, the number of pulses needed to obtained a given error scales as $1/\eta^2$ when $\eta \to 0$.

The intuition is straightforward. Because every optical and classical choice is randomized, a malicious receiver's strategy boils down to bookkeeping: count how many 0-, 1-, 2-, ...-photon pulses arrived, then decide how many in each bucket to acknowledge. With that reduction in hand, one can easily check that the linear combination $c'P - cP'$ is independent of whatever the receiver does with the 2-photon bucket—remember P and P' are the acknowledged counts for the two intensities, while c and c' are the corresponding Poisson weights of 2-photon events. Removing the dominant loophole from the single-intensity setting of [DKL12], namely the free play on two-photon pulses, forces the adversary to rely on much rarer higher-photon events. That is exactly what yields the tighter bounds and the improved $1/\eta^2$ scaling in the low-transmittance regime.

8.4 Verification with Weak Coherent Pulses

The full SDQC protocol needs no new security proof once the WCP-based preparation is in place. We simply plug the composable building block in:

1. Run the dummyless VBQC protocol.
2. Every time the verifier would call the RSP resource as the sender, substitute the multi-intensity WCP procedure (Protocol 10) followed by the gadget to distill a single hidden qubit.

Abstract Cryptography then does the rest: the construction error of the RSP block adds to the global correctness and security bounds. No further simulator needs to be written, and no trap analysis has to be redone.

One warning remains: we have closed the side-channel opened by multi-photon event, yet we have assumed no other side-channels are present in the WCPGenerator. In the event a device exposes other side channels they must be addressed separately.

8.5 Takeaways

In this chapter, we have gained practicallity and insights. We transformed weak off-the-shelf optics into a clean resource. Insisting on composability, we obtain a verification protocol that:

- avoids single-photon sources and still verifies BQP computations.
- scales as $1/\eta^2$—for two intensities—instead of $1/\eta^4$, a practical gain over long fibres.
- keeps the heavy lifting on the protocol side (randomisation, estimation, privacy amplification), not on the hardware.

There are trade-offs though. The gadget adds complexity, and pushing the scaling further—to the $1/\eta$ of folklore decoy proofs—would likely require more intensities and an even hairier analysis.

In addition, the same machinery immediately strengthens QKD proofs: it fixes a gap in decoy-state security arguments and delivers full composability.

In short: careful modularisation plus statistics let us recycle today's QKD infrastructure for tomorrow's verified quantum cloud.

Part IV
Extending Capabilities

This chapter explores two extensions to VBQC protocol. The first one is turning single-verifier protocols into secure multi-party delegated quantum computations. The development of Secure Multiparty Quantum Computation was chronologically the first project that was started. It lasted for a long time with ups and downs as we even broke an earlier version of the protocol, realizing at that occasion that blindness does not prevent attacks on specific types of qubits. We also better understood that attacker that is allowed to intervene in between trusted step rather than at the end was, even for perfectly blind protocols, is a much more adverse situation from the point of view of the honest party. While this project was started early, it is only after introducing our modular verification framework [KKL+24] that we managed to advance in the right direction. Indeed, it is for the purpose of this project that we introduced the dummyless protocol, and realized that the gadget from [DKL12] could serve many purposes. To avoid repetition, we will present the lift from SMPC to SMPQC by building on the already presented dummyless verification protocol.

The second extension is about securely delegating fault-tolerant quantum computations. The problem lies in the failure of the naive solution consisting of running entire protocols encoded into an error-correcting code. Indeed, even if one is willing to grant the verifier to perform single logical-qubit preparations, security remains the challenge. The reason for such difficulty is that instead of using a 2-dimensional system, it is replaced by a high-dimensional system whose extra dimension increase the attack surface for a malicious party. This was acknowledged in detail in [ABOEM17]. The solution we propose starts by properly defining imperfections—called compromise events—which are assumed to be stochastic. From there, it adds requirements to the fault-tolerance design rules in order to ensure not only correctness—standard fault-tolerance—but also security, whenever the compromise probability is below a constant threshold. This is then used to concretely construct the logically encoded RSP, thereby clearing the path to practical implementations of statistically secure, large-scale verified quantum computation.

9 Secure Multiparty Quantum Computation

This chapter shows how to lift a classical secure multiparty computation to the quantum world within our modular VBQC framework. We keep the hardware asymmetric——one quantum server, many lightweight clients—and the security composable. The core steps are:

- Replace the single client VBQC private traps and pads by collective ones, so as to accommodate multiple parties without revealing where traps and data live.
- Introduce a purely classical orchestrator to aggregate randomness, choose traps, and drive the UBQC dialogue without touching any qubits.
- Realize the orchestrator's only quantum need—remote state preparation—via a Collective RSP primitive that a single honest client is enough to secure.
- Finally, discharge the orchestrator by running a standard classical SMPC among the clients, preserving the global security bound.

The result is a statistically secure lift of a classical SMPC into SMPQC for BQP computations, achieved with minimal quantum trust and network complexity.

9.1 Motivation

Classical secure multiparty computation (SMPC) has been around for decades and is now a mature toolbox: a set of parties can jointly evaluate a function of their private inputs while learning nothing beyond their own output. Secure multiparty quantum computation (SMPQC) predates client–server verification: already in the early 2000s, quantum protocols were proposed to let several distrustful parties securely evaluate a joint circuit without revealing their inputs (see e.g. [CGS02, BOCG$^+$06, DNS12, DGJ$^+$20]).

Our aim here is different in flavor. We want to keep the VBQC asymmetry—one quantum workhorse, many lightweight clients—while moving from a single client to many. Doing this in MBQC raises two immediate questions:

1. How do we hide all sensitive structure—pads, traps, inputs—when no single client can be trusted with it?
2. How do we remain secure if an arbitrary subset of clients colludes with the server?

Answering these will occupy the rest of the chapter: we first identify what truly needs to be shared, then show how to share it without breaking blindness or verifiability, and finally fold everything back into a composable security statement.

9.2 Challenges for Security in Multi-Party Scenario

We adopt the strongest natural threat model: any party marked dishonest (server or client) may behave in an arbitrarily malicious, stateful, and adaptive way. Dishonest parties may collude without restriction—exchanging all their classical transcripts and any quantum side information they hold. Under such a model it is always sound to merge the colluding set into a single, all-powerful adversary. By contrast, honest clients cannot be safely merged. They do not know who else is honest and must not assume shared secrets.

With that in mind, a useful maneuver is to study extreme cases by aggregation. One such case is where all clients are honest, but the server dishonest. If we fuse the honest clients into one super-client, we should recover a textbook single-client VBQC: random $\mathsf{Z}(\theta)$ pads hide inputs; traps catch deviations.

Split that super-client back into individual ones, however, and two problems surface.

- *Ownership of traps is fatal.* Traps must cover every site where a harmful deviation could occur. If a trap—or its one-time pad—is owned by client j, then downstream dependencies will reveal where the client's data or trap lay—violating blindness.
- *Selective cheating becomes possible.* The colluding block (server + any dishonest clients) knows which qubits it helped encrypt. It also knows which ones it did not. Deviations can then target the latter, eroding the detection/insensitivity guarantees that underpin single-client VBQC security proofs.

Hence pads and traps cannot be private possessions. They must be collective objects—prepared and encrypted so that no strict subset of honest players—and certainly not the adversary—can tell which qubits are traps or carry data. As a consequence of the traps covering all the possible locations, data will also need to be collectively encoded to keep the indistinguishability between the two.

9.3 Recovered Security through Collective RSP

The previous section told us what we need: traps and pads that belong to no single client. Here we show how to manufacture them.

The first trick is to introduce a new party, the orchestrator, who will own the prepared quantum state, but without having to have quantum capabilities on his own. The second trick is already an old one: it relies on the phase-adding gadget of [DKL12]. If several parties each supply a random equatorial qubit, then—with the help of a round of CNOT's and measurements in the computation basis—their phases simply add up. One honest client is enough to hide the sum. That is exactly the collective one-time pad we require.

We formalize this as a resource and a simple protocol that constructs it, where several Alices play the clients, Bob the server, and Oscar the orchestrator.

Resource 8. Collective Remote State Preparation for n Senders

Inputs:

- Oscar has as input an angle $\theta \in \Theta = \left\{\frac{k\pi}{4}\right\}_{k\in\{0,\ldots,7\}}$.
- The Alices have no input.
- Bob has no input.

Computation by the Resource: The Resource prepares and sends the state $|+_\theta\rangle$ to Bob.

The protocol for constructing this resource is then:

Protocol 11. Collective Remote State Preparation

Input: Oscar has as input an angle $\theta \in \Theta$. Alices and Bob have no input.
Protocol:

- Alice number j samples $\theta_j \leftarrow\!\!\$\ \Theta$ and sends $\left|+_{\theta_j}\right\rangle$ to Bob.
- Alice j sends θ_j to Oscar using a Secure Classical Channel.
- For each $j \neq n$, Bob applies $\mathsf{CNOT}_{n,j}$ between the qubits n and j, with the first being the control and the second the target. It measures the target qubit j in the computational basis with measurement outcome t_j. It sends the vector $\boldsymbol{t}$ containing all the measurement outcomes to Oscar.
- Oscar computes $\theta' = \theta_n + \sum_{j\in[n-1]}(-1)^{t_j}\theta_j$ and sends a correction $(b, (-1)^b\theta - \theta')$ to Bob with $b \leftarrow\!\!\$\ \{0,1\}$. Bob then applies $\mathsf{X}^b\mathsf{Z}((-1)^b\theta - \theta')$ to the unmeasured qubit, keeping it as output.

In effect, the protocol allows Oscar to implement an RSP without having any quantum capabilities.

Theorem 13. Security of Collective Remote State Preparation

Providing a CRSP with the possibility to hide the preparation of dummy qubits is difficult. Indeed, early papers exploring RSP [Looo, LS03] point toward impossibility. More precisely, if we prepare dummies and states in the XY plane, we prepare a /generic/ ensemble of states in the language defined in [LS03]. By requiring security, we have that any such protocol can be transformed into an equivalent one in which the sender also performs operations that are fixed in advance. But if we provide the sender a fixed state, say $|+\rangle$ this cannot be the case. So we don't seem to be able to perform the operation—the link is that we want each of them to apply an encryption on top of an already encrypted state... but that seems to be possible if we restrict to OTP encoding by performing teleportation...

Protocol 11 perfectly constructs the Remote State Preparation Resource 3 from Secure Classical Channel Resources between each Client and the Orchestrator, for malicious coalitions that include the Server and at most $n-1$ Clients.

9.4 Verification with Collective RSP

The CRSP primitive lets a classical orchestrator drive VBQC while the clients supply only random equatorial qubits and their chosen classical inputs. We first spell out that orchestrated version. Then we remove the orchestrator: a classical SMPC among the clients emulates its role without altering the global security bound.

Orchestrated SMPQC

Protocol 12. Secure Multi-Party Delegated Quantum Computation with Ortchestrator

Public Information:

- $G = (V, E, I, O)$, a graph with input and output vertices I and O respectively;
- $\{I_j\}_{j\in[n]}$, a partition of the input vertices, with each I_j being associated to Alice number j.
- $\boldsymbol{P}$, a trappified scheme on graph G;
- $\preceq_G$, a partial order on the set V of vertices;
- N, d, w, parameters representing the number of runs, the number of computation runs, and the number of tolerated failed tests.

Alices' Inputs:

- Alice number j has as input a classical bit-string $x_j \in \{0,1\}^{|I_j|}$.
- The n Alices collectively have as input a set of angles $\{\phi_i\}_{i\in V}$ and a flow f which induces an ordering compatible with $\preceq_G$.

Protocol:

1. All Alices send their input x_j to Oscar, together with the computation angles $\{\phi_i\}_{i\in V}$ and flow f. Let x be the concatenation of all x_j.
2. Oscar and Bob perform an execution of the Trappified Blind Delegated Quantum Computation (Protocol 5). Instead of having Oscar send rotated states during the UBQC execution, it performs for each state an instance of Protocol 11 implementing CRSP together with the n Alices.
   (a) Oscar samples uniformly at random a subset $C \subset [N]$ of size d representing the computation runs.
   (b) For $k \in [N]$:
      i. If $k \in C$, Oscar sets the computation for the run to $(\{\phi_i\}_{i\in V}, f)$ with input x. Otherwise, Oscar samples a test (T, σ, τ) from the trappified scheme $\boldsymbol{P}$.
      ii. Oscar and Bob execute the chosen run with the UBQC Protocol 2. For each qubit sent during the execution of the protocol, they instead execute the Collective RSP Protocol 11 together with the n Alices. Oscar samples the Collective RSP with input and angle θ each time sampled at random.
      iii. If the run is a test, Oscar checks whether it passed.
   (c) If the number of failed tests is greater than w, Oscar sets the output to $(\perp, \mathsf{Abort})$.
   (d) Otherwise, let O be the majority vote on the output results of the computation runs. Oscar sets the output to (O, Accept).
3. Oscar sends the output to all Clients.

Removing the Orchestrator

In the previous protocol, the orchestrator's actions are purely classical. They can be replaced by a classical SMPC among the clients. It needs only to provide composable security to perform coin-tossing, basic string operations (array lookup) and computations in $\mathbb{Z}_8$ and $\mathbb{Z}_2$.

The security of this fully distributed SMPQC is given by the following theorem:

Theorem 14.

Suppose that the Trappified Blind Delegated Quantum Computation (Protocol 5) ϵ_V-constructs the Secure Delegated Quantum Computation with classical-I/O (Resource 2) for leak l_ρ. Then Protocol 12 ϵ_V-constructs the Quantum Secure Multi-Party Computation with classical-I/O from an interactive Classical Secure Multi-Party Computation for the same leak l_ρ, against malicious coalitions that include at most the Server and $n-1$ Clients.

The idea of the proof is simple. It is enough to retrace the order of compositions and use abstract cryptography to keep track of each construction error. All being at most negligible, and each resource being call at most a polynomial number of times, the security error remains negligible.

9.5 Takeways

We have shown that there is no fundamental difference in capabilities between the symmetric setup and the asymetric one presented here. Both can be used to lift a composable classical SMPC to the quantum realm. To this end, we have used that one honest client already suffices to hide every phase and every trap. The heavy classical bookkeeping—who randomized what, where the traps sit, how to drive UBQC—can be concentrated in a purely classical orchestrator, and once written that way it can just as well be emulated by a standard composable SMPC. The modular framework earns it here: we swap RSP for CRSP, the single-client for an orchestrator and then for SMPC, and the global security bound never has to be re-derived.

The asymmetry of resources is preserved: the server does the quantum work, clients remain light, and the network need not be complete. There are, however, natural limits. Our construction is tailored to BQP computations; pushing quantum data through the same pipeline would require traps that live outside a single Bloch-plane or a different masking mechanism altogether. Architectures such as QLine [PLL+23] suggest that even the CRSP step can be realized in hardware, by letting phases accumulate along a single fiber.

In short, MBQC matches circuit-based approaches for BQP computation, and gains a practical path to implementation. The remaining open question—collective RSP beyond one plane—are where the next advances could be.

10 Secure Delegated Fault-Tolerant Quantum Computation

This chapter tackles the question all earlier protocols politely dodged. Can a verification scheme survive the avalanche of physical noise that accumulates in a large, fault-tolerant quantum computation? Our goal is to show that it can. We extend the modular VBQC framework to the fault-tolerant regime, and we do so without sacrificing composability nor efficiency—security error is still negligible in physical resources.

In short, the chapter demonstrates that verifiable delegation scales: one can delegate a full fault-tolerant BQP computation, keep the verifier's secrets, and retain a composable security bound negligible in the size of the logical registers. What we do not claim is that the verifier can stay limited to physical single-qubit operations—our construction grants the verifier single logical-qubit operations. Whether this can be lifted remains open.

10.1 Motivation

Shrinking the verifier's quantum duties was worth the effort—proof-of-concept experiments, certification, and cost all benefit from it. Yet one question still looms: do these protocols scale?

As they stand, they do not. The best we achieved so far is robustness to global noise: if only a fixed fraction of whole rounds misfire, repetition and majority vote rescue the answer. Real devices misbehave differently. Errors strike locally, gate by gate. In a circuit with thousands of locations, even a per-gate error rate of 10^{-3} guarantees that almost every round suffers a non-trivial fault. Round-level repetition is then powerless. What we need is robustness at the gate level—in other words, a way to delegate fault-tolerant quantum computations securely.

Our route will therefore be: (i) make the imperfections explicit by modeling preparations, gates and measurements as stochastically compromised—their control parameters may trickle out, and individual subsystems may be handed to the adversary; (ii) rebuild Remote State Preparation at the logical level of a concatenated code with transversal $\mathsf{Z}(\theta)$ gates; (iii) split-compile each secret rotation into two random pieces and follow it by two error-correction blocks, so the effective leak probability drops quadratically at every concatenation step; and (iv) plug this fault-tolerant RSP back into the dummyless VBQC template and prove a threshold-style theorem [AGP06]: below a constant noise rate, the adversary's distinguishing advantage stays negligible even as the depth grows.

10.2 Side Channels Due to Error Correction

In Chapter 7 we learned a lesson: to keep blindness intact, and thus verifiability, any deviation before a secret gate must be pushable to after that gate without picking up dependence on the secret. Allowing the prover to supply the state and letting the verifier apply a hidden $\mathsf{Z}(\theta)$ was fatal precisely because commuting an arbitrary error through a secret rotation imprints the secret angle θ on the error.

Exactly the same mechanism lurks for fault-tolerance. In the naive logical RSP construction, replace the single-qubit by a logical block: we prepare a logical $|+\rangle$, then apply a logical $\mathsf{Z}(\theta)$. Suppose the code is CSS and admits transversal $\mathsf{Z}(\theta)$.

A single physical X error on one qubit of the block, followed by a perfect logical $Z(\theta)$, behaves differently depending on θ: if $\theta = 0$, the error stays X; if $\theta = \pi/2$, it turns into Y. The syndrome then changes character: from purely Z-type it becomes mixed X/Z-type. Should the subsequent error correction miss that fault, the prover upon receiving the prepared faulty logical $|+_\theta\rangle$ can, by reading out the syndrome, glean information about θ.

One "if" too many? Security arguments are exactly about preventing long chains of "ifs" from aligning.

This example calls for a precise model of the imperfections that might arise on the verifier's side. This model must be compatible with standard fault-tolerance yet reflect delegation. To this end, we switch terminology for describing the model, where we have a user trying to perform quantum operations and an eavesdropper that plays the role of the adversary interferring with the computation. In textbook fault-tolerance [AGP06], the eavesdropper is typically granted the full circuit description even before fault locations are fixed. In a delegated setting, this would instantly kill secrecy. To avoid that, we distinguish between public parameter domains and the chosen value. Preparations, gates, and measurements are parameterized: the set Λ is public, but the actual $\lambda \in \Lambda$ leaks only stochastically, while the eavesdropper can then choose an error to apply on the corresponding physical qubit. This is captured by:

Resource 9. Stochastically Compromised Preparations, Gates and Measurements

A *Stochastically Compromised Preparation, Gate or Measurement* with compromise probability p_c is an ideal resource that behaves in the following way, where Alice plays the user and Bob the eavesdropper:

Public Information:

- The compromise probability p_c;
- The size of the input and output registers n;
- The set of parameters that can be used to classically control the operation Λ, reduced to the name of the preparation, gate or measurement if not parametrized;
- The set of CPTP maps $\{U(\lambda)\}_{\lambda\in\Lambda}$.

Alice's Input:

- A value $\lambda \in \Lambda$ that parametrizes the preparation, gate or measurement;
- The quantum registers that are acted upon by the Resource.

Bob's Input: Bits $c_i \in \{0, 1\}$ indicating whether it wants to cheat, set to 0 in the honest case and to 1 when it wants to compromise the i^{th} location of the preparation, gate or measurement.

Computation by the Resource:

- If the input register is non-empty, it inserts the state ρ of the provided register into its internal register. Else it initializes its input register in the $|0\rangle^{\otimes n}$ state.
- It applies $U(\lambda)$ to ρ.
- For each i s.t. $c_i = 1$, with probability p_c:
  - It sends λ and the register at location i to Bob;
  - It receives from Bob a quantum system and inserts into its internal register at position i.
- It outputs the state in its internal register at Alice's output interface.

This resource faithfully reproduces the informal problem above. Implement the naive logical RSP with compromised $Z(\theta)$ gates, and every physical site becomes a potential leak of θ. As the block size grows, the probability that some site spills the secret tends to one. We therefore need an RSP construction that explicitly withstands stochastic compromise events. In the next section we show how concatenation and angle splitting drive the effective compromise rate down to negligible as the size of logical registers increases.

10.3 Secure Fault-Tolerant RSP via Split-Compilation

Conditions for Privacy and Gadget Construction

The first constraint for our design is that the preparation has to be fault-tolerant. This does not necessarily imposes that everything on the verifier's side is fault-tolerant—the prover could possibly assist with multi-qubit gates and error correction—but we were not able to maintain security in this scenario. We thus resorted to grant the verifier the ability to prepare and operate on a single logical-qubit by itself. It would then follow the naive implementation proposed above: prepare a logical $|+\rangle$ and apply a logical $\mathsf{Z}(\theta)$. To keep the circuit uniform over all θ, we pick a code with a transversal $\mathsf{Z}(\theta)$: the $[[15,1,3]]$ quantum Reed–Muller code. This avoids magic-state injection—whose circuitry would depend on θ and could thus introduce side-channels—and keeps the proof tractable.

The second constraint is that compromise probability must not scale with the block size. As we have seen, if each physical $\mathsf{Z}(\theta)$ is compromised with probability p_c, a transversal gate leaks almost surely once the block is large enough. We fix this by ensuring that the effective probability of compromise events is squared for each added concatenation level. The trick is to apply *split-compilation*: to add one level of concatenation, each physical qubit is transformed into a code-block of 15 qubits; the $Z(\theta)$ is applied on each wire i by sampling $\alpha_i \leftarrow\!\!\$\ \Theta$, setting $\beta_i = \theta - \alpha_i$, and applying $\mathsf{Z}(\alpha_i)$ followed by $\mathsf{Z}(\beta_i)$. Intuitively, both must be compromised for the adversary to reconstruct θ.

Yet, this intuition is only partly correct. A single incoming X error can still drag θ into the syndrome if the error-correction block is itself compromised. The fix is simple: after each logical gate we run two EC blocks. If one is compromised, the other resets the syndrome to the all-zero string and forgets about θ.

This yields the level-1 split-compilation for $\mathsf{Z}(\theta)$—below 0-Ga is synonym for physical gate, 1-Ga are gates applied to logical qubits encoded using one level of concatenation:

Definition 13. 1-Safe$\mathsf{Z}(\theta)$

Using the $[[15,1,3]]$ quantum Reed Muller code, the $1-\mathsf{SafeZ}(\theta)$ gate that implements the corresponding 1-Ga for the single-qubit $\mathsf{Z}(\theta)$ gate is constructed as follows:

For $i \in [1,15]$:
- Sample $\alpha_i \leftarrow\!\!\$\ \Theta$.
- Set $\beta_i = \theta - \alpha_i$.
- Apply in succession the two physical gates corresponding to $\mathsf{Z}(\beta_i) \circ \mathsf{Z}(\alpha_i)$ to the i^{th} qubit, via calls to the Stochastically Compromised Gates Resource.

This gives the following procedure for the recursive simulation:

Definition 14.

A 1-*SafeRec* for simulating a 0-Ga consists of the 1-Ga followed by two consecutive 1-EC for each output block of the 1-Ga.

Here, 1-EC is the level-1 error-correction routine that corrects a single logical-qubit encoded using one level of concatenation.

Definition 15. k-SafeRec

A k-SafeRec is obtained from a $(k-1)$-SafeRec by replacing each 0-Ga by its corresponding 1-SafeRec.

For avoiding confusion, we use the terminology of accurate and private so that it is distinct from the correct and secure ones that we reserve for the constructed RSP. Indeed, here, we will improve the accuracy and privacy by concatenation, up to a level where it will be good enough to provide correctness and security of the corresponding RSP. These notions of accuracy and privacy need not compose as they are only used in the context of the verifier-side RSP, while correctness and security do compose at the RSP level.

Security of Fault-Tolerant RSP

While the robustness to noise of the fault-tolerant RSP follows the usual threshold machinery, privacy does not come for free. We must rework the argument so that, after simulation, (i) the logical action is *accurate* and (ii) *private* in the sense that the entire syndrome history is independent of the secret angles θ. The main result is:

Theorem 15. Threshold theorem for accurate and private computations

The diamond distance δ between a computation with L locations simulated using k levels of concatenation and an ideal computation performing the correct computation on the logical space and producing a syndrome that is independent of the parameters θ of the gates satisfies

$$\delta \leq 2L \times p_0 (p_c / p_0)^{2^k}, \tag{10.1}$$

where p_c is the compromise event probability of Resource 9.

The proof mirrors the classical threshold proof [AGP06], but each step now has a security twin:

- *Level-0 to level-1 improvement.* Show that accuracy and privacy improve when moving from level-0 to level-1, for sufficiently low compromise probability.
- *Extended Safe Rectangles.* As in [AGP06], we enlarge rectangles—a k-Ga followed by the two \$$k$-ECs—to include one preceding k-EC. Here this is needed not only for accuracy—to catch incoming errors—but also to take into account the possibility of inducing θ-dependence by commutation.
- *Good vs. bad recursion.* We define conditions where simulations would be either inaccurate or non-private. This notion of goodness and badness is essential to perform the recursion and extend the quadratic decrease of effective compromise probability from level-0 to level-1.
- *Bad rectangles as compromised gates.* The technical part is again ensuring that a bad level-k rectangle can be represented as a simulated compromised gate.

A remark in [AGP06] is instructive: "the syndrome becomes an effective environment that [...] interacts with the data during o-faults. In their setting this is harmless because errors can be arbitrarily correlated; in ours it becomes a side channel affecting the secrecy of the computation. The second EC block precisely severs this channel: unless both blocks are compromised, one of them resets the syndrome to all zeros and erases any trace of θ in the syndrome.

Armed with this theorem, we construct a level-k simulation of RSP and provide its security guarantee with Alice playing the role of the user and Bob that of the eavesdropper:

Protocol 13. Level-k Simulation of Remote State Preparation

Alice: Choose $\theta \in \Theta$.
Alice: Implements the level-k preparation of $|+\rangle$.
Alice: Implements the level-k $Z(\theta)$ using the Safe$Z(\theta)$ construction.
Bob: For each location in the circuit decide to cheat, i.e. stochastically compromise it, and if succeeding gets the leaks and provides the replacement for the leaked registers.
Alice: Sends the produced quantum state.

Theorem 16.

Protocol 13 constructs the Level-k Single Plane Remote State Preparation—i.e. the Single-Plane RSP (Resource 3) applied on level-k logical qubits with error bounded above by $4p_0(p_c/p_0)^{2^k}$.

10.4 Verification of Fault-Tolerant Quantum Computations

The full protocol is obtained by taking the dummyless verification template and replacing every call to RSP by its level-k fault-tolerant version. Gates and measurements on the prover's side are likewise lifted to their level-k simulations.

Protocol 14. Secure Delegation of Fault-Tolerance Quantum Computation

Public Information:

- $G = (V, E, I, O)$, a graph with input and output vertices I and O respectively.
- $\mathfrak{W}$, the set of dummyless tests on graph G.
- $\preceq_G$, a partial order on the set V of vertices.
- N, d, w, parameters representing the number of runs, the number of computation runs, and the number of tolerated failed tests.

- k, the concatenation level used to encode logical information.

Alice's Inputs: A set of angles $\{\phi(i)\}_{i\in V}$ and a flow f which induces an ordering compatible with $\preceq_G$.
Protocol: Alice and Bob execute Protocol 5 with dummyless traps for the intended pattern on graph $G = (V, E)$ in the following way:

- For each call to the RSP, they run Protocol 13.
- For each gate or measuremenent instructed by Alice, Bob implements the corresponding level-k simulation, considering for measurements that the outcome is that for the measurement of the corresponding logical qubit.

The overhead and the security bound follow from simple counting plus composability.

Theorem 17. Threshold Theorem for Secure Delegation of Fault-Tolerant Quantum Computations

Let d be proportional to N, and let c the fixed bounded error of the BQP *class of computations. Let ϵ be the (constant) lower bound on the probability that a test sampled uniformly from set $\mathfrak{W}$ fails in the presence of a* Z *error. Let w be the maximum number of test rounds allowed to fail, chosen such that $w < \frac{2c-1}{2c-2}(N-d)(1-\epsilon)$. Let the leak be defined as $l_{\mathsf{U}} = (G, \mathfrak{W}, \preceq_G)$, and let the Verifier use a register of size 15^k corresponding to k level of concatenation of the quantum [15,1,3] Reed-Mueller code with $15^k \in O(Nlog(|V|))$. Let p_c be the probability that a location is compromised, and p_0 be the threshold probability from Eq.* (10.1).

Then, Protocol 14 $\eta(N)$ constructs the Secure Delegated Quantum Computation (Resource 2) in the Abstract Cryptography framework for $\eta(N)$ negligible in N.

10.5 Takeaways

We have shown how to remove the main scalability roadblock, but one sharp question survives: can the verifier drop back to single physical-qubit operations and still stay secure? Yet, this looks largely a theoretical question. Without its own error correction, a verifier must lean on the prover for every correction step, forcing interactive, back-and-forth rounds between logical gates. That chatter would explode the latency and sink performance.

In short, we now know how to scale securely—but doing so with a truly qubit-only verifier remains an open, and probably impractical, challenge.

Part V
Conclusions & Perspectives

This work, begun in 2019 and developed throughout this manuscript, turns verification from a single monolithic proof into a stack of interoperable modules. By cleanly separating blindness, trappification, embedding, and fault-tolerance, we could patch specific shortcomings of earlier protocols and push the field in two complementary directions:

- Lighter clients & lower overhead for servers.
- Richer functionality, from multi-party inputs to gate-level noise robustness.

The broader aim is practical: bring verification and blindness into the design brief of hardware vendors, cloud providers, and end-users—before devices reach the too big to secure-by-design regime. Showing that the same template handles secure multi-party quantum computation and tolerates physical noise at the gate level is a first step toward that integration.

But the road is long, and several milestones lie ahead.

Perspectives

Verification is still viewed by many hardware vendors as an after-thought, something to bolt on once the hardware works. Experience with classical security—and with quantum error correction—suggests the opposite: early design choices lock in what becomes possible later. In particular, adaptivity and the central role of a secure RSP must be recognised up-front. Quantum analogues of trusted-execution environments could fill the gap where high-bandwidth quantum links are impractical, but such isolation shapes the rest of the QPU stack.

Our view is that vendors will embrace verification once it offers clear, immediate value to them. Two levers can help.

- *Benchmark-driven competition.* Trap-based tests give a worst-case to average-case reduction, linear-time scaling, and negligible security error per extra resource invested. Turning these tests into public benchmarks should spark vendorcompetition and push verification features into the control stack.
- *In-field characterisation.* Trap outcomes double as a fine-grained log of faults. A server that knows it is honest can mine those logs to calibrate and stabilize its device without downtime. Early inclusion of verification thus pays off even before quantum advantage is reached.

Action plan

The first priority is to translate every module of the framework into the circuit model, so that superconducting platforms can adopt the techniques without resorting to MBQC work-arounds. In parallel, we should refine the mathematical analysis until we can obtain hardware efficient proofs. Finally, the initial trap-based benchmarks must be carried through to public pilot reports, turning them into yardsticks that vendors cannot ignore.

Next, we will explore how far quantum communication can be reduced by exposing a tightly quantified amount of information about the delegated algorithm, or by leaning on standard computational assumptions. At the same time, we need to lift robustness from majority-vote classical outputs to genuine quantum outputs encoded in error-correcting codes. Alongside those technical goals, we will package the protocol suite into an open-source library that hides cryptographic plumbing from circuit designers yet remains auditable by specialists.

Long-term questions

Over the longer horizon, we should seek a resource-theoretic account of verification that reveals which primitives are truly fundamental. We must also confront the long-standing challenge of achieving statistical soundness with a fully classical client—no qubits, no measurement device—while keeping complexity reasonable. Finally, we ought to clarify what it really means to "verify a quantum output" once

the output itself is an entangled state destined for further quantum processing, and to devise tests that respect that richer notion of correctness.

Part VI
Appendix

A Abstract Cryptography

The Abstract Cryptography (AC) security framework [MR11, Mau12] used in this work follows the *ideal/real simulation paradigm*. A protocol is considered secure if it is a good approximation of an ideal version called a *resource*. Its main advantage is that any system that follows the structure defined by the framework is inherently composable, in the sense that if two protocols are secure separately, the framework guarantees at an abstract level that their sequential or parallel execution is also secure. We refer the reader to [DFPR14] for a more in-depth presentation, in particular regarding the framework's composability in the context of SDQC.

In this framework, the purpose of a secure protocol π is, given a number of available resources $\mathcal{R}$, to construct a new resource – written as $\pi\mathcal{R}$. This new resource can be itself reused in a future protocol. A resource $\mathcal{R}$ is described as a sequence of CPTP maps with an internal state. It has *input and output interfaces* describing which party may exchange states with it. It works by having each party send it a state (quantum or classical) at one of its input interfaces, applying the specified CPTP map after all input interfaces have been initialized and then outputting the resulting state at its output interfaces in a specified order. An interface is said to be *filtered* if it is only accessible by a dishonest player. The actions of an honest player i in a given protocol is also represented as a sequence of efficient CPTP maps π_i – called the *converter* of party i – acting on their internal and communication registers. We focus here on the two-party setting, in which case $\pi = (\pi_1, \pi_2)$.

In order to define the security of a protocol, we need to give a pseudo-metric on the space of resources. We consider for that purpose a special type of converter called a *distinguisher*, whose aim is to discriminate between two resources $\mathcal{R}_1$ and $\mathcal{R}_2$, each having the same number of input and output interfaces. It prepares the input, interacts with one of the resources according to its own (possibly adaptive) strategy, and guesses which resource it interacted with by outputting a single bit. Two resources are said to be indistinguishable if no distinguisher can guess correctly with good probability.

Definition 16. Statistical Indistinguishability of Resources

Let $\epsilon > 0$, and let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two resources with same input and output interfaces. The resources are *ϵ-statistically-indistinguishable* if, for all unbounded distinguishers $\mathcal{D}$, we have:

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

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

The construction of a given resource $\mathcal{S}$ by the application of protocol π to resource $\mathcal{R}$ can then be expressed as the indistinguishability between resources $\mathcal{S}$ and $\pi\mathcal{R}$. More specifically, this captures the correctness of the protocol. The security is captured by the fact that the resources remain indistinguishable if we allow some parties to deviate in the sense that they are no longer forced to use the converters defined in the protocol but can use any other CPTP maps instead. This is done by removing the converters for those parties in Equation A.1 while keeping only $\pi_H = \prod_{i \in H} \pi_i$ where H is the set of honest parties. The security is formalized as follows in Definition 17 in the case of two parties.

Definition 17. Construction of Resources

Let $\epsilon > 0$. We say that a two-party protocol π ϵ-statistically-constructs resource $\mathcal{S}$ from resource $\mathcal{R}$ if:

1. It is correct: $\pi\mathcal{R} \underset{\epsilon}{\approx} \mathcal{S}$.
2. It is secure against malicious party P_i for $i \in \{1,2\}$: there exists a *simulator* (converter) σ_i such that $\pi_j\mathcal{R} \underset{\epsilon}{\approx} \mathcal{S}\sigma_i$, where $j \neq i$.

The General Composition Theorem (Theorem 1 from [MR11]) captures the security of protocols which use other secure protocols as subroutines, in sequence or in parallel.

Theorem 18. General Composability of Resources

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

- *The protocols are* sequentially composable*:*

$$\alpha\mathcal{R} \underset{\text{stat, } \epsilon_\alpha}{\approx} \mathcal{S} \;\wedge\; \beta\mathcal{S} \underset{\text{stat, } \epsilon_\beta}{\approx} \mathcal{T} \;\Rightarrow\; (\beta \circ \alpha)\mathcal{R} \underset{\text{stat, } \epsilon_\alpha+\epsilon_\beta}{\approx} \mathcal{T}. \tag{A.2}$$

- *The protocols are* context-insensitive*:*

$$\alpha\mathcal{R} \underset{\text{stat, } \epsilon_\alpha}{\approx} \mathcal{S} \;\Rightarrow\; (\alpha \mid \mathsf{id})(\mathcal{R} \mid \mathcal{T}) \underset{\text{stat, } \epsilon_\alpha}{\approx} (\mathcal{S} \mid \mathcal{T}). \tag{A.3}$$

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

Bibliography

[ABOE10] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computation. In *Proceedings of Innovations of Computer Science (ICS 2010)*, page 453–469, 2010.

[ABOEM17] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive proofs for quantum computations. *arxiv:1704.04487*, 2017. Updated and corrected version of arxiv:0810.5375.

[AGP06] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *Quant. Inf. Comput.*, 6:97–165, 2006.

[AV13] Dorit Aharonov and Umesh V. Vazirani. *Is Quantum Mechanics Falsifiable? A Computational Perspective on the Foundations of Quantum Mechanics*, page 329–350. The MIT Press, June 2013.

[BCC+20] Christian Badertscher, Alexandru Cojocaru, Léo Colisson, Elham Kashefi, Dominik Leichtle, Atul Mantri, and Petros Wallden. *Security Limitations of Classical-Client Delegated Quantum Computing*, page 667–696. Springer International Publishing, 2020.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In IEEE, editor, *50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

[BKMP07] Daniel E Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9(8):250, 2007.

[BOCG+06] Michael Ben-Or, Claude Crepeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 249–260, Washington, DC, USA, 2006. IEEE Computer Society.

[Bro18] Anne Broadbent. How to verify a quantum computation. *Theory of Computing*, 14(11):1–37, 2018.

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multiparty quantum computation. In *Proceedings of the Thiry-fourth Annual ACM Symp. on Theory of Computing*, STOC '02, page 643, New York, NY, USA, 2002. ACM.

[DCEL09] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80(1), July 2009.

[DFPR14] Vedran Dunjko, Joseph F. Fitzsimons, Christopher Portmann, and Renato Renner. Composable security of delegated quantum computation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 406–425, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[DGJ+20] Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 729–758, Cham, 2020. Springer International Publishing.

[DK06] Vincent Danos and Elham Kashefi. Determinism in the one-way model. *Phys. Rev. A*, 74:052310, Nov 2006.

[DKL12] Vedran Dunjko, Elham Kashefi, and Anthony Leverrier. Universal blind quantum computing with weak coherent pulses. *Phys. Rev. Lett.*, 108(200502), 2012.

[DKP07] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *J. ACM*, 54(2), April 2007.

[DNS12] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Actively secure two-party evaluation of any quantum operation. In *Advances in Cryptology–CRYPTO 2012*, pages 794–811. Springer, 2012.

[FHcvM18] Joseph F. Fitzsimons, Michal Hajdušek, and Tomoyuki Morimae. Post hoc verification of quantum computation. *Phys. Rev. Lett.*, 120:040501, Jan 2018.

[FK17] Joseph F. Fitzsimons and Elham Kashefi. Unconditionally verifiable blind quantum computation. *Phys. Rev. A*, 96:012303, Jul 2017.

[FKD18] Samuele Ferracin, Theodoros Kapourniotis, and Animesh Datta. Reducing resources for verification of quantum computations. *Physical Review A*, 98(2):022323, 2018.

[FKD19] Samuele Ferracin, Theodoros Kapourniotis, and Animesh Datta. Accrediting outputs of noisy intermediate-scale quantum computing devices. *New Journal of Physics*, 21(11):113038, nov 2019.

[GKW15] Alexandru Gheorghiu, Elham Kashefi, and Petros Wallden. Robustness and device independence of verifiable blind quantum computing. *New Journal of Physics*, 17(8):083040, 2015.

[GLMO24] Maxime Garnier, Dominik Leichtle, Luka Music, and Harold Ollivier. Composably secure delegated quantum computation with weak coherent pulses. In *2024 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, page 221–225. IEEE, July 2024.

[GLMO25] Maxime Garnier, Dominik Leichtle, Luka Music, and Harold Ollivier. Multi-pulse protocol for securely preparing a single (qu)bit (in preparation), 2025.

[HKSE17] D Hangleiter, M Kliesch, M Schwarz, and J Eisert. Direct certification of a class of quantum simulations. *Quantum Science and Technology*, 2(1):015004, feb 2017.

[JWH+19] Yang-Fan Jiang, Kejin Wei, Liang Huang, Ke Xu, Qi-Chao Sun, Yu-Zhe Zhang, Weijun Zhang, Hao Li, Lixing You, Zhen Wang, Hoi-Kwong Lo, Feihu Xu, Qiang Zhang, and Jian-Wei Pan. Remote blind state preparation with weak coherent pulses in the field. *Physical Review Letters*, 123(10), September 2019.

[Kap16] Theodoros Kapourniotis. *Efficient verification of universal and intermediate quantum computing*. PhD thesis, School of Informatics, University of Edinburgh, 2016.

[KDK15] Theodoros Kapourniotis, Vedran Dunjko, and Elham Kashefi. On optimising quantum communication in verifiable quantum computing, 2015. Presented at AQIS'15 conference; arXiv:1506.06943.

[KKL+24] Theodoros Kapourniotis, Elham Kashefi, Dominik Leichtle, Luka Music, and Harold Ollivier. Unifying quantum verification and error-detection: Theory and tools for optimisations. *Quantum Science and Technology*, 9(3):035036, May 2024.

[KKL+25] Theodoros Kapourniotis, Elham Kashefi, Dominik Leichtle, Luka Music, and Harold Ollivier. Asymmetric quantum secure multi-party computation with weak clients against dishonest majority. *Quantum Science and Technology*, 10(2):025015, 2025.

[KLMO24] Elham Kashefi, Dominik Leichtle, Luka Music, and Harold Ollivier. Verification of quantum computations without trusted preparations or measurements. *CoRR*, 2024.

[KLMO25] Theodoros Kapourniotis, Dominik Leichtle, Luka Music, and Harold Ollivier. Plugging leaks in fault-tolerant quantum computation and verification, 2025.

[KP17] Elham Kashefi and Anna Pappa. Multiparty delegated quantum computing. *Cryptography*, 1(2):1–20, 7 2017.

[KW17] Elham Kashefi and Petros Wallden. Optimised resource construction for verifiable quantum computation. *Journal of Physics A: Mathematical and Theoretical; preprint arXiv:1510.07408*, 2017.

[LMKO21] Dominik Leichtle, Luka Music, Elham Kashefi, and Harold Ollivier. Verifying BQP computations on noisy devices with minimal overhead. *PRX Quantum*, 2:040302, Oct 2021.

[Loo0] Hoi-Kwong Lo. Classical-communication cost in distributed quantum-information processing: A generalization of quantum-communication complexity. *Physical Review A*, 62(1), June 2000.

[LS03] Debbie W. Leung and Peter W. Shor. Oblivious remote state preparation. *Physical Review Letters*, 90(12), March 2003.

[Mah18] Urmila Mahadev. Classical verification of quantum computations. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 259–267. IEEE Computer Society, 2018.

[Mau12] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Theory of Security and Applications*, pages 33–56, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[MF16] Tomoyuki Morimae and Joseph F. Fitzsimons. Post hoc verification with a single prover. *CoRR*, 2016.

[MKA+22] Yao Ma, Elham Kashefi, Myrto Arapinis, Kaushik Chakraborty, and Marc Kaplan. QEnclave – a practical solution for secure quantum cloud computing. *npj Quantum Information*, 8(1):128, 2022.

[MPS22] Mehdi Mhalla, Simon Perdrix, and Luc Sanselme. Shadow pauli flow: Characterising determinism in mbqcs involving pauli measurements. *CoRR*, 2022.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *Innovations in Computer Science*, pages 1 – 21. Tsinghua University Press, jan 2011.

[NV17] Anand Natarajan and Thomas Vidick. A quantum linearity test for robustly verifying entanglement. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC '17. ACM, June 2017.

[PLL+23] Beatrice Polacchi, Dominik Leichtle, Leonardo Limongi, Gonzalo Carvacho, Giorgio Milani, Nicolò Spagnolo, Marc Kaplan, Fabio Sciarrino, and Elham Kashefi. Multi-client distributed blind quantum computation with the qline architecture, 2023.

[RB01] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.

[RUV13] Ben W. Reichardt, Reichardt Falk Unger, and Umesh Vazirani. Classical command of quantum systems. *Nature*, 496:456–460, 2013.

[Sim21] Will Simmons. Relating measurement patterns to circuits via pauli flow. *CoRR*, 2021.