# Optimised resource construction for verifiable quantum computation

OPEN ACCESS
IOP Publishing
Journal of Physics A: Mathematical and Theoretical
J. Phys. A: Math. Theor. **50** (2017) 145306 (25pp)
https://doi.org/10.1088/1751-8121/aa5dac

# Optimised resource construction for verifiable quantum computation

**Elham Kashefi**[1,2] **and Petros Wallden**[1]

[1] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom
[2] Departement Informatique et Reseaux, CNRS LTCI, Telecom ParisTech, Paris CEDEX 13, France

E-mail: petros.wallden@ed.ac.uk

Received 6 September 2016, revised 12 December 2016
Accepted for publication 2 February 2017
Published 8 March 2017


**Abstract**

Recent developments have brought the possibility of achieving scalable quantum networks and quantum devices closer. From the computational point of view these emerging technologies become relevant when they are no longer classically simulatable. Hence a pressing challenge is the construction of practical methods to verify the correctness of the outcome produced by universal or non-universal quantum devices. A promising approach that has been extensively explored is the scheme of verification via encryption through blind quantum computation. We present here a new construction that simplifies the required resources for any such verifiable protocol. We obtain an overhead that is linear in the size of the input (computation), while the security parameter remains independent of the size of the computation and can be made exponentially small (with a small extra cost). Furthermore our construction is generic and could be applied to any universal or non-universal scheme with a given underlying graph.

Keywords: scalable quantum network, quantum devices, blind quantum computation

(Some figures may appear in colour only in the online journal)

## 1. Introduction

It is widely believed that quantum computers, and quantum devices in general, can outperform their classical counterparts. There are problems which can be solved efficiently by quantum computers that is believed classical computers require an exponentially (in the size of the input) long time for. If the problem is in NP, a classical verifier can efficiently check the result of the quantum device. However, there are problems believed to be outside NP, such as quantum simulation [1, 2] or other BQP problems [3] that the verifier needs to resort to different techniques to detect a 'dishonest' quantum device. Currently the most efficient way to verify a quantum computation is to employ cryptographic methods, where we have an almost classical verifier that executes the computation using an untrusted but fully quantum prover. There have been a number of such verification methods [4–20] where generally there exists a trade-off between the practicality of the scheme versus their generality, trust assumptions and security level. It is the target of this work to both reduce the experimental requirements of the most general schemes and to achieve further improvements in the more restricted schemes. In general, in order to make quantum verification schemes practical a number of different aspects have been considered. While a full review of those aspects is beyond the scope of this paper it is worth noticing that most of them have been addressed using protocols based on verification via blind quantum computing [4, 6–8, 13–15]. We will refer to this family of protocols collectively as verifiable blind quantum computation (VBQC) schemes, where the key idea is based on hiding the underlying computation (also known as blindness). This would allow the verifier to encode simple trap computations within a general computation that runs on a remote device in such a way that the computation is not affected, while revealing no information to the device. The correctness of the general computation is then tested via the verification of the trap computation. The latter is significantly less costly and thus leads to an efficient scheme (essentially similar to an error detection code). What makes the procedure work is the blindness which hides the trap computation from the actual one. To elaborate further on the security parameter scaling, consider the following informal definition of *verification* that we formalise later (for details see also [4]).

**Definition 1.** A quantum computation protocol is $\epsilon$-verifiable if the probability of accepting an incorrect output for any choice of the prover's strategy is bounded by $\epsilon$.

In a practical scenario, to be convinced of the correctness of the output obtained from a given quantum device, one needs a verification protocol where the security parameter ($\epsilon$) can be made arbitrarily small while keeping the cost (in terms of the experimental requirements) optimal. The standard technique for amplification when dealing with classical output is to simply repeat the protocol multiple times (let us say $d$) and if all rounds are accepted and result in the same outputs, then this output is correct except with probability $\epsilon^d$. However, dealing with quantum output requires more elaborate methods (to deal with the possibility of coherent attacks) which involves the use of full fault-tolerant computation and the presence of multiple traps in order to achieve exponential bounds.

It is evident that in order to obtain a *verifiable* quantum computation, some extra cost in terms of resources is needed. However, one wishes to ensure that the extra cost of verification is not excessive, and in particular that it is not more than the speed-up that one obtains from using a quantum algorithm. Here it is worth mentioning, that quantum algorithms, in many cases, provide polynomial speed-up (e.g. Grover's search[3]) and if their verification requires extra quadratic cost it could reduce considerably or even annihilate the advantage of the quantum algorithm.

[3] Note however, that in the specific case of search algorithms, they belong to NP, and thus are classically verifiable without the need for a quantum verification protocol.

### *1.1. Our contribution*

In this work we focus to further improve the underlying resource construction required for VBQC schemes. Our main results can be summarised as follows:

(i) In section 2, inspired by the dotted-complete graph state introduced in [4], we give a generic construction where for any given (universal or non-universal graph state resource) multiple trap qubits isolated from the computation qubits can be added. Unlike the dotted-complete graph state the overhead of the new construction is only linear (instead of quadratic) in the size of the specific computation that will be performed. Furthermore the traps are uniformly distributed and their positions are essentially independent from each other.

(ii) We use this construction to obtain a new universal VBQC protocol (section 3) that has a lower cost. Since we are using a different resource, the proof technique had to be adapted accordingly. Our protocol even before adding any boosting mechanism has a constant security parameter and thus allows a straightforward one-shot experiment.

(iii) When the output of the quantum computation is classical, we use a repetition technique to boost the security of our protocol to arbitrarily small $\epsilon$ (section 4.1). Importantly, we can achieve this using a constant number of repetitions which is independent of the size of the computation and scales with the desired security parameter leading to an overall cost $O(\log \frac{1}{\epsilon}) \times O(N)$. In previous VBQC protocols the number of repetitions that were required increased with the size of the computation.

(iv) For the general quantum output case, we use a fault-tolerant encoding of the computation to boost the security to arbitrary small $\epsilon$ while at the same time we still requiring only a linear, in the size of the computation, overhead (section 4.2) with a moderate extra cost that depends on the security parameter and scales as $O(\log \frac{1}{\epsilon})$ depending on the fault-tolerant encoding used. The overhead of previous VBQC protocols (except [7]) is quadratic (on top of the security-parameter logarithmic dependency).

Our construction could be used to optimise various other existing VBQCs (see appendix F).

### *1.2. Related works*

There have been a number of papers on verification addressing different aspects. We do not aim to give a complete list but we give here a brief description of some related works. Aharonov, Ben-Or and Eban [5] provided the first verification protocol. This required a linear overhead in the size of the computation, but also required a verifier that has involved quantum abilities, in particular the ability to prepare entangled states of size which depend on the security parameter.

Following another approach, based on measurement-based quantum computation, Fitzsimons and Kashefi [4] obtained the most optimal scheme from the point of view of the capability of verifiers. However, the overhead of the full scheme becomes quadratic. Recently a solution for addressing this issue was proposed in [7] by combining the above two protocols [4, 5]) to construct a hybrid scheme. This was the only verification protocol (before our work) that requires a linear number of qubits while at the same time requires that the verifier has the minimal quantum property of preparing single quantum systems. However, the protocol requires the preparation of qudits (rather than qubits) where the dimension is dictated by the desired level of security. Moreover the required resource is still constructed based on a dotted-complete graph state but of small constant size. Hence further investigation is needed to compare the experimental simplicity of the two schemes, ours and the one in [7].

The first experimental implementation of a simplified verification protocol was presented in [6] where a repetition technique was also explored. Other experiments on verifiable protocols include [19] and an experiment based on the protocol in [17]. However, none of these works are applicable to a full universal scheme such as ours.

On the other hand, to achieve a classical verifier new techniques are proposed either using two provers at the cost of increasing the overall overhead of the protocol dramatically [11] or increasing the number of the provers further [12]. Other device-independent protocols [13, 14] used a single universal quantum prover and an untrusted qubit measuring device and while the complexity improved (compared to the two provers protocol [11]) it was still far from experimentally realisable.

The VBQC protocol could be generally viewed as a prepare and send scheme (using the terminology from quantum key distribution). Equivalent schemes based on measurement-only could also be obtained [9, 10]. In this scenario the prover prepares a universal resource and sends it qubit-by-qubit to the verifier that performs different measurements in order to complete a quantum computation. These protocols are referred to as online protocols (in contrast to the offline protocols mentioned above) since the quantum operations of the verifier occur when they know what they want to compute. The online scheme can also achieve verification either by creating traps [9], or by measuring the stabiliser of the resource state [10]. These protocols could be improved using our techniques (see appendix F).

Finally a composable definition of [4] is given in [15], while a limited computational model (one-pure-qubit) is examined in [8]. Due to the generic nature of our construction these results would also be applicable to our protocol.

The verification protocols in [16, 20] are teleportation based. Due to the general mapping (see [21, 22]) between the teleportation (with two-qubit measurement) and one-way computation (with single-qubit measurement), one can also explore any possible improvement that our techniques could bring to these new protocols. For example, it may be possible to amplify the probability of success for quantum output with minimal extra cost, given a constant probability of error of the 'one-shot' protocol (which is already achieved in [16]) combining the technique of [4] that uses fault-tolerant encoding with our local resource construction.

### *1.3. Background*

The family of VBQC protocols are conveniently presented in the measurement-based quantum computation (MBQC) model [23] that is known to be the same as any gate teleportation model [24]. We will assume that the reader is familiar with this model; further details can be found in [22]. The general idea behind an MBQC protocol is that one starts with a large and highly entangled multiparty state (the resource state) and the computation is performed by carrying out single-qubit measurements. There is an order to the measurements since the basis of a measurement may depend on outcomes of previous measurements. The resource states used are known as *graph states* as they can be fully determined by a given graph see details in [25]. One way to construct a graph state given the graph description is to assign to each vertex of the graph a qubit initially prepared in the state $|+\rangle$ and for each edge of the graph to perform a controlled $-Z$ gate to the two adjacent vertices.

If one starts with a graph state where qubits are prepared in a rotated basis $|+_\theta\rangle = 1/\sqrt{2}\,(|0\rangle + \mathrm{e}^{\mathrm{i}\theta}|1\rangle)$ instead, then it is possible to perform the same computation with the non-rotated graph state by performing measurements in a similarly rotated basis. This observation led to the formulation of the *universal blind quantum computation* (UBQC) protocol [26] which hides the computation in a client-server setting. Here a client prepares rotated qubits, where the rotation is only known to them. The client sends the qubits to the server (as

soon as they are prepared, hence there is no need for any quantum memory). Finally the client instructs the server to perform entangling operations according to the graph and perform single qubit measurements in suitable angles in order to perform the desired computation (where an extra randomisation $r_i$ of the outcome of the measurements is added). During the protocol the client receives the outcomes of previous measurements and can classically evaluate the next measurement angle. Due to the unknown rotation and the extra outcome randomisation, the server does not learn what computation they actually perform.

The UBQC protocol can be uplifted to a verification protocol where the client (referred to now as verifier) can detect a cheating server (referred to now as prover). To do so, the verifier for certain vertices (called dummies) sends states from the set $\{|0\rangle, |1\rangle\}$ which has the same effect as a $Z$-basis measurement on that vertex. In any graph state if a vertex is measured in the $Z$-basis it results in a new graph where that vertex and all its adjacent edges are removed. During the protocol the prover does not know for a particular vertex if the verifier sends a dummy qubit or not. This enables the verifier to isolate some qubits (disentangled from the rest of the graph). Those qubits have fixed deterministic outcomes if the prover follows the instructions honestly. The positions of those isolated qubits are unknown to the prover and the verifier uses them as traps to test that the prover performs the quantum operations that is given. This technique led to the first universal VBQC protocol [4] which is the basis of our paper. While the trapification idea is straightforward, it is challenging to find the optimal way of inserting trap qubits while not breaking the general computation. This is the central focus of this paper: to introduce a general optimised scheme for constructing graph state resources for VBQC protocols.

## 2. The dotted triple-graph construction

Our construction starts with a 'base' graph $G$ such that the related graph state $|G\rangle$ can be used as the resource to perform a particular (or universal) quantum computation in MBQC. This graph is then 'decorated' in a suitable way, resulting to a graph that we will call dotted triple-graph $DT(G)$ that defines the resource state $|DT(G)\rangle$ for running a verified quantum computation in an efficient way. The general idea is to construct the $DT(G)$ graph which after some operations (chosen secretly by the verifier) can be broken to three identical graphs. One will be used to perform the desired computation and the other two to insert trap computations to detect possible deviations. The way that the $DT(G)$ is broken is chosen by the verifier and thus the prover is blind to which vertex belongs to which graph. This general idea was first introduced in [4]. The key difference of our construction is that while in [4] the breaking to subgraphs occurs in a global way, in our construction it happens locally. This difference results in a reduction on the number of vertices (and thus qubits).

Our local construction, defined precisely later, means that the prover can obtain certain information about the graph without compromising the security. Therefore knowledge or leaking of secret parameters at one part of the computation does not affect other positions. This property makes the present construction particularly useful for applications and extensions that involve multiple parties, a fact exploited in the secure two-party quantum computation protocol of [27].

In this section we will only give definitions and properties of the dotted triple-graph construction when viewed purely as graph operations. These properties will play a crucial role in the next sections where we will use as a resource state the dotted triple-graph state $|DT(G)\rangle$ in order to obtain verifiable quantum computation protocols.

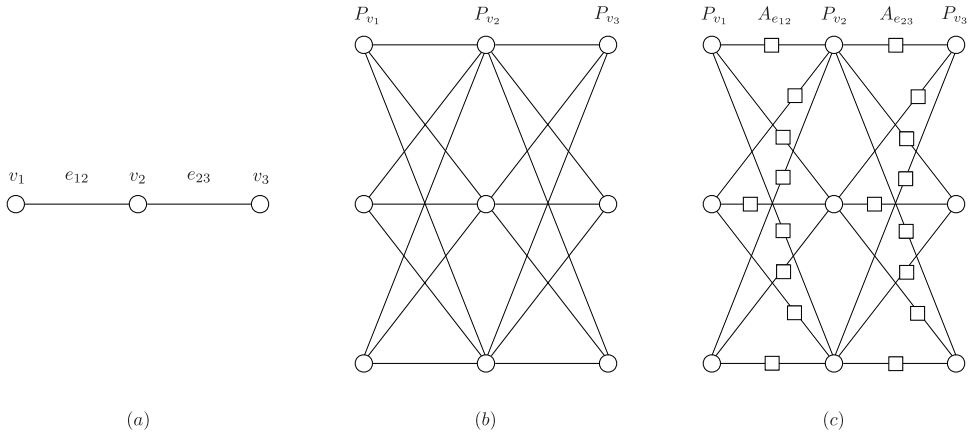


**Figure 1.** (a) A base graph consisting of three vertices and two edges. (b) A triple-graph $T(G)$ where for each vertex $v$ there is a set of three vertices $P_v$. (c) A dotted triple-graph. For each edge of the base graph there is a set of nine added vertices $A_e$. The added vertices are denoted as squares, while the primary as circles.

**Definition 2 (Introduced in [4]).** We define the *dotting* operator $D$ on graph $G$ to be the operator which transforms a graph $G$ to a new graph denoted as $D(G)$ and called *dotted* graph, by replacing every edge in $G$ with a new vertex connected to the two vertices originally joined by that edge[4]. We call the set of vertices of $D(G)$ previously inherited from $G$ primary vertices $P(D(G))$, and the new vertices as added vertices $A(D(G))$.

### *2.1. Dotted triple-graph construction*

(i) We are given a base graph $G$ that has vertices $v \in V(G)$ and edges $e \in E(G)$, as in figure 1(a). In the following steps we will give the new graph $DT(G)$, called dotted-triple graph and specify its vertices and edges.
(ii) For each vertex $v_i$, we define a set of three new vertices $P_{v_i} = \{p_1^{v_i}, p_2^{v_i}, p_3^{v_i}\}$.
(iii) Corresponding to each edge $e(v_i, v_j) \in E(G)$ of the base graph that connects the base vertices $v_i$ and $v_j$, we introduce a set of nine edges $E_{e(v_i,v_j)}$ that connect each of the vertices in the set $P_{v_i}$ with each of the vertices in the set $P_{v_j}$.
(iv) The graph that its vertices are $\bigcup_{v_i \in V(G)} P_{v_i}$ and the edges are defined as in the previous step is called triple-graph $T(G)$, as in figure 1(b).
(v) We perform the dotting operator $D$ on the triple graph $T(G)$ to obtain the dotted triple-graph $DT(G)$. An example of dotted triple-graph can be seen in figure 1(c).

Note that, according to definition 2 and the labelling in the above construction the primary vertices are given as $P(DT(G)) = \cup_{v_i} P_{v_i}$. For convenience we also label the added vertices $A(DT(G))$ as follows. Corresponding to each edge $e(v_i, v_j)$ of the base graph $G$, there are now nine added vertices and we will denote each set of added vertices as $A_{e_{ij}} = \{a_1^{e_{ij}}, \cdots, a_9^{e_{ij}}\}$. Note that the number of vertices of the new graph is $|V(DT(G))| = 3|V(G)| + 9|E(G)|$. If the maximum degree of the base graph is a constant $c$ then the number of vertices of the $DT(G)$ are linear in the number of vertices of the base graph. This property means that if we can base our verifiable quantum computation protocol on this graph, then the number of qubits we will need is linear in the size of the computation.

[4] The dotting operation is also known as edge subdivision.

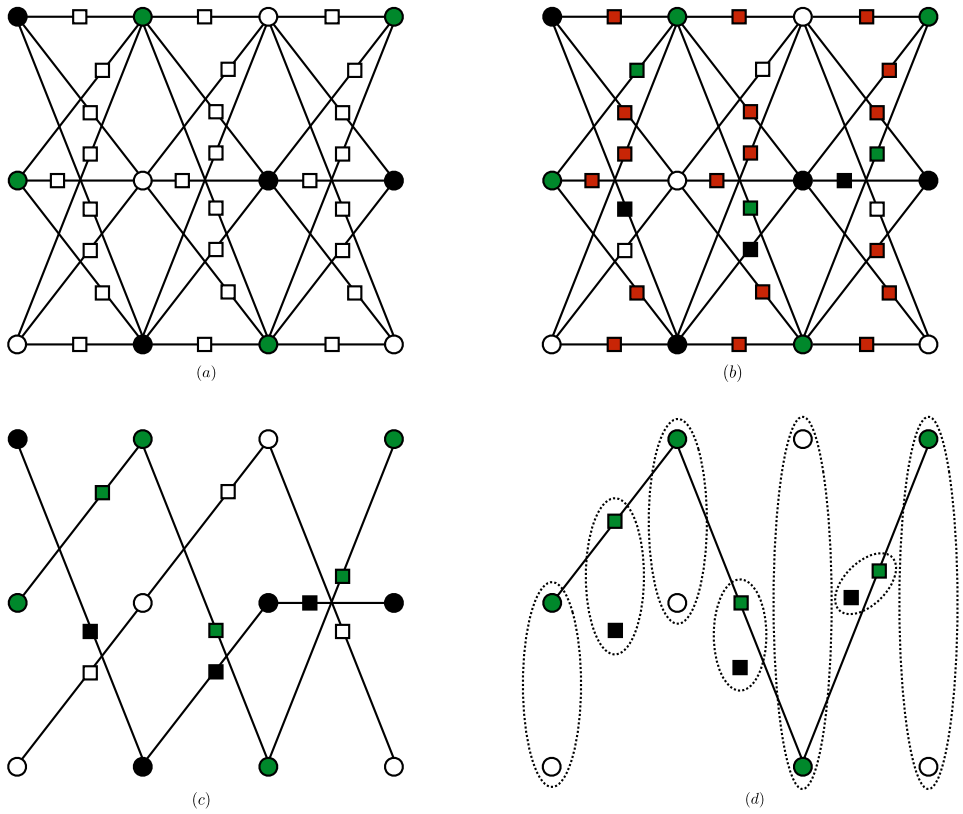


**Figure 2.** (a) A dotted triple-graph, where only the primary vertices are coloured, and this is done randomly for each set. (b) A trap-colouring of $DT(G)$ that is fully fixed from the colouring of the primary vertices. (c) $DT(G)$ after performing break operations on all red vertices. This results to three copies of the dotted base graph. (d) $DT(G)$ after performing further break operations on the primary vertices of the black graph and added vertices of the white graph. The result is a dotted base graph (green) and isolated white traps on primary vertices and black traps on added vertices. For each green vertex there is a corresponding trap. (a) Primary vertices coloured independently. (b) Trap-colouring. (c) Three dotted base graphs after breaking the red vertices. (d) Computation graph and isolated while and back traps.

In what follows we present a labelling scheme that for convenience we present as a colouring (however, connected vertices could be the same colour).

**Definition 3 (Trap-Colouring).** We define trap-colouring to be an assignment of one colour to each of the vertices of the dotted triple-graph that is consistent with the following conditions.

(i) Primary vertices are coloured in one of the three colours of white, black or green.
(ii) Added vertices are coloured in one of the four colours of white, black, green or red.
(iii) In each primary set $P_v$ there is exactly one vertex of each colour.
(iv) Colouring the primary vertices fixes the colours of the added vertices: added vertices that connect primary vertices of different colour are red. Added vertices that connect primary vertices of the same colour get that colour.

Note that the choice of colours for each of the primary sets $P_v$ can be chosen randomly and is independent from the choices made on other primary sets. We can also see that in each of

the added sets $A_e$ we have one white, one black, one green and six red vertices. It is easy to see that such labelling can be obtained efficiently for any graph.

In figures 2(a) and (b) we see an example of trap-colouring, where in (a) we choose independently the colour choices of primary vertices and in (b) the colours of added vertices is fixed following the rules for trap-colouring given above.

**Definition 4 (Introduced in [4]).** We define the *break* operator[5] on a vertex $v$ of a graph $G$ to be the operator which removes vertex $v$ and any adjacent edges to $v$ from $G$.

**Lemma 1.** *Given the dotted triple-graph DT(G) and a trap-colouring, by performing break operations on the red vertices we obtain three identical copies*[6] *of the dotted base graph D(G) each of them consisting of a single colour.*

The proof is given in appendix A.1. Figure 2(c) illustrates how the $DT(G)$ breaks to three identical dotted base graph, after performing break operations on the red vertices.

**Definition 5.** We define the *base-location* of a vertex $f \in V(DT(G))$ of the dotted triple-graph to be the position that the set $P_v$ or $A_e$ that includes $f$ has in the dotted base graph $D(G)$. This position is denoted by either '$v$' corresponding to the specific primary vertex of $D(G)$ or with '$e$' corresponding to the specific added vertex of $D(G)$ on the edge $e$.

Given a trap-colouring, each primary vertex belongs to one of the three graphs where the colour is determined by the trap-colouring. However, its base-location is fixed prior to the trap-colouring. Here we can see the difference of our construction with that of [4]. There a dotted-complete graph was used and the graph also broke to three identical graphs, where all primary vertices belonged to one of these graphs. However, there was no restriction as to how this break happens, and any choice of three equal subsets was valid. In our construction we maintain the structure of the base-location (reducing the number of added vertices required), but at the same time the colour choices at one primary base-location are totally independent from colour choices at other primary base-location.

**Lemma 2.** *Given a dotted graph D(G), by applying break operators to every vertex in P(D(G)) or A(D(G)) the resulting graph is composed of the vertices of A(D(G)) or P(D(G)) respectively and contains no edges.*

This property was essentially proved in [4] (see appendix A.2) and is required for the verification protocols presented in the next sections. In figure 2(d) we see after the break operations of figure 2(c), further break operations performed on all white added vertices and on all black primary vertices. We end up with a (green) copy of the dotted base graph and white isolated traps at primary vertices and black isolated traps at added vertices.

There are common properties that we will prove for both primary and added vertices and for the ease of notation we will refer to either such set $P_v$ or $A_e$ as $F_l$ with the convention that the subscript $l$ denotes the base-location of the set and when it takes value $v$ (primary base-location) it becomes $P_v$ and when it takes value $e$ (added base-location) it becomes $A_e$.

Next we show that while the trap-colouring is a global construction it can indeed be considered as a local scheme. This property will be explored in our proof technique for the verification. We formalise this notion in the next set of definitions and lemmas.

**Definition 6.** We define *local-colouring* of a set $F_l$ to be an assignment of colours to that set that is consistent with some global trap-colouring.

[5] The break operator is also known as vertex deletion.
[6] Also known as isomorphic graphs.

This definition captures the idea of colouring a particular set $F_l$ corresponding to base-location $l$ such that it can be part of some global trap-colouring without having *any* further constraints from colours of vertices at other base-locations. We can see that a local-colouring of an added set $A_{e_{ij}}$ fully determines the colours of the vertices in the two neighbouring primary base-locations $P_{v_i}, P_{v_j}$, while the converse is also true. A local-colouring of the two primary sets $P_{v_i}, P_{v_j}$ fully determines the colours of the added set $A_{e_{ij}}$. We can therefore see that a local-colouring of set $A_{e_{ij}}$ is equivalent with a local-colouring of the two neighbouring primary base-locations $P_{v_i}, P_{v_j}$. We can also see that a local-colouring of *all* primary sets $P_v$ is compatible with a trap-colouring and fixes it uniquely.

However, if we have two general sets $F_{l_1}, F_{l_2}$ it is not always possible to colour them both using a local-colouring and still be able to find a global trap-colouring. An example is if we have a primary set $P_{v_1}$ and its added neighbouring set $A_{e_{1k}}$, where a local colouring of the set $P_{v_1}$ imposes constraints on the colours of $A_{e_{1k}}$ further than those required from a local-colouring. E.g. an added vertex connected to a white primary vertex can be either white or red, but can never be black. An added set $A_{e_{ij}}$ can have local-colouring if there is no constraint on the colours from the neighbouring primary sets $P_{v_i}, P_{v_j}$, but also from other added sets $A_{e_{ik}}, A_{e_{jk}}$ that have common neighbour sets (either $P_{v_i}$ or $P_{v_j}$). We wish to make it very clear that when there is a collection of base-locations that one can assign (independently) local-colourings to all the related sets $F_l$ and still be able to always find a global trap-colouring.

**Definition 7 (Independently colourable locations (ICL)).** Given a dotted triple-graph $DT(G)$ and a collection of $n$ base-locations $\mathcal{E}$ with corresponding sets $F_l$, we call the set $\mathcal{E}$ *independently colourable locations* if any local-colouring of the sets $F_l$ is consistent with at least one trap-colouring.

We should stress at this point, that ICL is a property of a collection of *base-locations* and not of vertices. The motivation is that for those base-locations, one could independently colour the vertices of each base-location and obtain an allowed trap-colouring. In other words, what this definition captures is that the choice of colours within each of the sets $F_l$ corresponding to a base-location in $\mathcal{E}$ is independent from the choice of colours in other sets $F_{l'}$ with base-location in $\mathcal{E}$.

For each base-location $l$ we define $\epsilon_l = \{l\}$ if the base-location is primary and $\epsilon_l = N_{D(G)}(l)$ if the base-location is added (i.e. in the latter case, it contains the two primary base-locations that are adjacent to the location $l$).

**Lemma 3.** *A set of n base-locations $\mathcal{E}$ is ICL if and only if for all pairs $i, j \in \mathcal{E}$ the sets $\epsilon_i \cap \epsilon_j = \varnothing$.*

The proof is given in appendix A.3. The following property is necessary for section 4.2.

**Theorem 1.** *Consider a dotted triple-graph DT(G). Consider a set S of n base-locations and assume that the base graph G has maximum degree c. Then there exists a subset $S' \subseteq S$ of these base-locations that are ICL (independent colourable locations) and it contains at least $|S'| = \frac{n}{2c+1}$ locations.*

**Proof.** The set $S$ has $n$ locations of the graph $D(G)$[7]. We want a subset of these locations $S'$ such that it satisfies lemma 3. The condition of that lemma requires that if a primary base-location $v_i$ is included, then all its neighbouring base-locations should be excluded. The maximum number of neighbours is given by the maximum number of added *base-locations* which is $c$. Therefore if we include the base-location $v_i$ in the set $S'$, we might need to exclude at most $c$ other base-locations from the set $S$.

[7] Note again, that here we are dealing with $D(G)$ and *not* $DT(G)$, and thus we are dealing with a set of base-locations and not of vertices of the $DT(G)$.

To include any added base-location $e_{ij}$ in the set $S'$, lemma 3 requires that its neighbours $v_i, v_j$ and the neighbours of its neighbours $e_{ik}, e_{jk}$ should be excluded. Its neighbours are 2, while the neighbours of the neighbours are at most $2(c-1)$. It follows that to include $e_{ij}$ in the set $S'$ we might need to exclude at most $2c$ other base-locations from the set $S$.

From the pigeonhole principle it follows that we can find a set $S'$ with at least $\frac{n}{2c+1} = |S'|$ base-locations that are ICL. □

The *existence* of this number of ICL is what is necessary for the proof of section 4.2. However, we should note that finding such $S'$ given $S$ can be done efficiently, essentially following the procedure of the above proof.

## 3. Verifiable quantum computation

We give a verifiable blind quantum computation protocol using the dotted triple-graph construction, but otherwise, we follow similar steps with [4]. With our construction we obtain a protocol where the probability of success is constant (independent of the size of the computation) and we use only linear, in the size of the computation, number of qubits.

As we mentioned in section 1.3 dummy qubits break the graph to the computation graph and isolated traps. This breaking is hidden from the prover, since the prover does not know the positions of dummy qubits. For the computation to be accepted, the prover needs to return the correct results for the isolated traps. In other words, a malicious prover that wants to deceive the verifier, needs in the same time to guess correctly all the traps *and* corrupt the computation by deviating on some of the computation graph qubits.

As it is evident from the protocol (see protocol 1), the positions of the dummy qubits (i.e. those that are $\{|0\rangle, |1\rangle\}$) is determined by the trap-colouring. It is easy to check that sending dummy qubits has the same effect as making a $Z$ measurement in MBQC which effectively breaks the graph state at this vertex. Therefore the properties defined in section 2 corresponding to the reduction of the $DT(G)$ to one dotted base graph $D(G)$ and isolated traps (lemmas 1 and 2) as well as the properties concerning the independence of the colouring and thus the distribution of traps (theorem 1), all apply here.

**Theorem 2 (Correctness).** *If both verifier and prover follow the steps of protocol 1 then the output is correct and the computation accepted.*

**Protocol 1.** Verifiable universal blind quantum computation using dotted triple-graph.

We assume that a standard labelling of the vertices of the dotted triple-graph $DT(G)$, is known to both the verifier and the prover. The number of qubits is at most $3N(3c+1)$ where $c$ is the maximum degree of the base graph $G$.

- **Verifier's resources**
  - Verifier is given a base graph $G$ that the dotted graph state $|D(G)\rangle$ can be used to perform the desired computation in MBQC with measurement pattern $\mathbb{M}_{\text{Comp}}$.
  - Verifier generates the dotted triple-graph $DT(G)$, and selects a trap-colouring according to definition 3 which is done by choosing independently the colours for each set $P_v$.
  - Verifier will send dummy qubits for all red vertices, and thus performs break operation.
  - Verifier chooses the green graph to perform the computation.
  - For the white graph verifier sends dummy qubits for all added qubits $a_w^e$ and thus generates white isolated qubits at each primary vertex set $P_v$. Similarly for the black graph the verifier sends dummy qubits for the primary qubits $p_b^v$ and thus generates black isolated qubits at each added vertex set $A_e$.
  - The dummy qubits position set $D$ is chosen as defined above (fixed by the trap-colouring).
  - A binary string $\mathbf{s}$ of length at most $3N(3c+1)$ represents the measurement outcomes. It is initially set to all zero's.

– A sequence of measurement angles, $\phi = (\phi_i)_{1 \leqslant i \leqslant 3N(3c+1)}$ with $\phi_i \in A = \{0, \pi/4, \cdots, 7\pi/4\}$, consistent with $\mathbb{M}_{\text{Comp}}$. We define $\phi'_i(\phi_i, \mathbf{s})$ to be the measurement angle in MBQC, when corrections due to previous measurement outcomes $\mathbf{s}$ are taken into account (the function depends on the specific base-graph and its flow, see e.g. [26]). We also set $\phi'_i = 0$ for all the trap and dummy qubits. The verifier chooses a measurement order on the dotted base-graph $D(G)$ that is consistent with the flow of the computation (this is known to prover). The measurements within each set $P_v, A_e$ of $DT(G)$ graph are ordered randomly.
– $3N(3c + 1)$ random variables $\theta_i$ with value taken uniformly at random from $A$.
– $3N(3c + 1)$ random variables $r_i$ and $|D|$ random variable $d_i$ with values taken uniformly at random from $\{0, 1\}$.
– A fixed function $C(i, \phi_i, \theta_i, r_i, \mathbf{s}) = \phi'_i(\phi_i, \mathbf{s}) + \theta_i + \pi r_i$ that for each non-output qubit $i$ computes the angle of the measurement of qubit $i$ to be sent to the prover.

**Protocol 2.** Cont. VUBQC using dotted triple-graph.

• **Initial Step**
– **Verifier's move:** Verifier sets all the value in $\mathbf{s}$ to be 0 and prepares the input qubits as

$$|e\rangle = X^{x_1} Z(\theta_1) \otimes \ldots \otimes X^{x_l} Z(\theta_l) |I\rangle$$

– $x_i$ are random variables with value uniformly at random from $\{0,1\}$.
The remaining qubits are prepared in the following form

$$\forall\, i \in D \qquad |d_i\rangle$$
$$\forall\, i \notin D \qquad \prod_{j \in N_G(i) \cap D} Z^{d_j} |+_{\theta_i}\rangle$$

and sends the prover all the $3N(3c + 1)$ qubits in the order of the labelling of the vertices of the graph.
– **Prover's move:** Prover receives $3N(3c + 1)$ single qubits and entangles them according to $DT(G)$.
• **Step** $i$ : $1 \leqslant i \leqslant 3N(3c + 1)$
– **Verifier's move:** Verifier computes the angle $\delta_i = C(i, \phi_i, \theta_i, r_i, \mathbf{s})$ and sends it to the prover.
– **Prover's move:** Prover measures qubit $i$ with angle $\delta_i$ and sends the verifier the result $b_i$.
– **Verifier's move:** Verifier sets the value of $s_i$ in $\mathbf{s}$ to be $b_i + r_i$.
• **Verification**
– After obtaining the output qubits from the prover, the verifier measures the output trap qubits with angle $\delta_t = \theta_t + r_t \pi$ to obtain $b_t$.
– Verifier accepts if $b_i = r_i$ for all the white (primary) and black (added) trap qubits $i$.
– Verifier applies corrections according to measurement outcomes $b_i$ and secret parameters $\theta_i, r_i$ at the output layer green qubits and obtains the final output.

**Proof sketch.** If both verifier and prover follow the steps of protocol 1 then the prover essentially (when pre-rotations are taken into account) applies the pattern $\mathbb{M}_{\text{Comp}}$ at the green dotted base graph $D(G)$, which by assumption performs the desired computation (see also theorems 1 and 3 of [4]). Moreover, the isolated white and black qubits are measured in the correct basis and thus the verifier receives $b_i = r_i$ for the traps and accepts the computation (for further details, see appendix B). □

As already stated, the protocol is $\epsilon$-verifiable if the probability of accepting an incorrect output for any strategy of the prover is bounded by $\epsilon$. We follow the same definitions as in [4], while for completion the exact meaning of 'strategy of prover' and expressions for 'incorrect output' and 'accepting' are given in appendix C.

**Theorem 3 (Verification).** *Protocol 1 is* $\left(\frac{8}{9}\right)$*-verifiable (for quantum or classical output).*

**Proof sketch.** The proof follows closely certain steps of the proof of theorem 8 of [4]. Here we give an outline of the proof (details in appendix C).

The aim is to show that the probability that a malicious prover corrupts the computation *and* succeeds in all traps (and thus the verifier accepts the output) is bounded by $\epsilon$. To achieve this we follow five steps. In **step 1** we prove that any deviation from the ideal protocol can be expressed in terms of some Kraus operators which are then written as linear combination of strings of Pauli matrices (denoted as $\sigma_i$) and the remaining of the proof is to see which of those attacks maximise the probability of accepting an incorrect outcome.

In **step 2** we note that there are some strings $\sigma_i$ that for *any* choice of the secret parameters (trap positions, angles, etc) of the verifier do not corrupt the computation and thus they do not contribute to the probability of failure. The set of all other strings $\sigma$ (that could corrupt the computation for some choice of parameters) will be denoted as $E_i$. It is clear that a malicious prover, to optimise the chance to get an incorrect outcome accepted, should only use attacks from the set $E_i$. In this section, where we consider the simplest protocol, a single non-trivial attack could corrupt the computation and the set $E_i$ consists of all the attacks $\sigma$'s that have in at least one position a non-trivial attack. However, in the next section this changes. The technique to amplify the success probability uses fault-tolerant encoding and thus the computation is corrupted only if multiple errors occur and this leads to different set $E_i$. For now we keep the description general for as long as possible, so that it applies for the next section. After the set $E_i$ is defined, in order to compute an upper bound for the failure probability, we simply compute the probability of not triggering any trap given that the attacks used are all from the set $E_i$. This is clearly an upper bound for the failure probability (worse-case scenario), since in reality the fact that there exist some choices of the secret parameters that a given $\sigma \in E_i$ corrupts the computation does not mean that it corrupts the computation in general. However, an upper bound $\epsilon$ of the failure probability suffices to prove that the protocol is $\epsilon$-verifiable.

In **step 3** we exploit the blindness of the malicious prover. The fact that they do not know the secret parameters restricts the attacks that contribute to attacks that are a convex combination of Pauli attacks. This is important since it eliminates the 'coherent' type of attacks and resembles theorems in quantum key distribution (QKD) that reduce coherent attacks to a collective by exploiting the symmetry of the states.

In **step 4** we show that a malicious prover maximises the value of the bound of failure probability if they perform an attack with exactly the fewest non-trivial attacks that are consistent with $E_i$ obtained from step 2. This is a single attack for the protocol of this section (but different in section 4.2). It is easy to see that the greatest value is obtained for a single $\sigma$. In the next steps of the proof we find the maximum value of our bound for an attack corresponding to the single optimal (for a malicious prover) $\sigma$.

Finally, in **step 5** we use the partition of the qubits to sets $P_v$ and $A_e$. It is important to note that within each of those sets there is exactly one computation qubit and exactly one trap qubit. From previous steps we know that the bound of the failure probability is highest if the malicious prover chooses to make a single non-trivial attack. This attack happens at a qubit that belongs to either some set $P_v$ or some set $A_e$. The probability of hitting a trap given a single set is clearly independent from the other free parameters corresponding to this qubit (but not the probability to detect it in general) and it goes as $1/|P_v|$ or $1/|A_e|$. This leads to a bound for the failure probability $\epsilon = 8/9$. □

## 4. Amplification of the probability of success

In the previous section we gave a simple construction to directly obtain a verification protocol with constant failure probability $\epsilon$. However, a verification protocol is successful if the $\epsilon$ of the failure probability can be made arbitrarily small. There are two techniques that have been

used to amplify the probability of success of a verification protocol. The first is simpler both conceptually and in terms of experimental requirements, but applies only in the case that the output of the quantum computation performed is classical. The second applies for computations with quantum output as well. We will use both techniques and show that starting with the dotted triple-graph construction we obtain in both cases improvements.

### *4.1. Amplification for classical output*

In the case that a quantum computation has a classical output (e.g. solving classical problems or sampling, etc) it suffices to have an $\epsilon$-verifiable protocol for *any* $\epsilon < 1$. This $\epsilon$ can be boosted and made arbitrarily small by repeating the protocol a sufficient number of times and accepting only when all repetitions agree. This results to an $\epsilon' = \epsilon^d$ which can be made as small as the security level required by choosing the number of repetitions $d$ suitably. This of course implies an extra communication cost, for the multiple copies prepared, which scales as $O(\log \frac{1}{\epsilon'})$ leading to overall complexity being $O(\log \frac{1}{\epsilon'}) \times O(N)$.

By using the dotted triple-graph construction we can obtain a repetition protocol where we only repeat a constant number of times (and the number of repetitions depends *only* on the security level). This is in contrast with the increasing number of repetitions needed in the repetition protocol used in [6] that was based on the brickwork-state protocol of [4]. It follows that the dotted triple-graph repetition protocol requires only a linear number of qubits. As we will see in the next section this does not give better complexity from the general protocol (that allows for quantum output). However, it has a number of practical advantages (easier to implement, smaller coefficient of leading term, etc) which can be of importance in view of the quantum systems that are being developed, such as Networked-Quantum-Information-Technologies (NQIT) [28]. Further details and an alternative construction applicable only for classical output can be found in appendix D.

### *4.2. Amplification for quantum output*

We now turn to the general case, where the output of the computation can be quantum. Our main result is that our DTG construction leads to an exponentially-secure verification protocol for quantum output with only a linear overhead. Similar to [4] we use a technique that encodes the computation in a fault-tolerant way in order to amplify the probability of success of the protocol. The particular size of the boosting achieved depends on the fault-tolerant code that is used. Here we treat the protocol in full generality.

The general idea is that the computation is encoded with fault-tolerant encoding, while the traps remain single (non-encoded) qubits. Therefore, while a single error on a trap leads to a rejection of the computation, to corrupt the actual output of the computation many errors on computation qubits are required. The malicious prover needs at the same time to avoid hitting any single trap *and* hit many computation qubits in order to corrupt the output.

---

**Protocol 3.** Boosted verifiable UBQC for quantum output, using dotted triple-graph and fault-tolerant encoding.

---

- Verifier chooses a base graph $G$ and a measurement pattern $\mathbb{M}_{\text{mathbbComp}}$ on the dotted base graph $D(G)$ that implements the desired computation in a fault-tolerant way, that can detect or correct errors fewer than $\delta/2$.
- **Verifier follows steps of protocol 1.**

---

**Theorem 4 (Verification).** *Protocol 3 is* $\left(\frac{8}{9}\right)^d$*-verifiable for quantum or classical output, where* $d = \lceil \frac{\delta}{2(2c+1)} \rceil$*, c is the maximum degree of the base graph and* $\delta$ *is the number of errors tolerated on the base graph G.*

**Proof sketch.** The proof follows similar steps to [4]. However, because of our local construction, the proof changes and we highlight here where our technique deviates. Since the computation is completed using a fault-tolerant encoding, any deviation that affects fewer than $\delta/2$ computation qubits does not corrupt the output. It follows that attacks which contribute to the $p_{\text{fail}}$ have non-trivial Pauli's in, at least, $\delta/2$ base-locations[8]. Here we used the fact that in our construction the prover knows the partition of the qubits with respect to their base-location and thus will necessarily attack at least $\delta/2$ base-locations since they wish to corrupt the computation. Using blindness (as in steps 3 and 4 of proof of theorem 3), we conclude that the fewer attacks (given that corruption is possible) maximises $p_{\text{fail}}$. According to our construction, in $\delta/2$ base locations, there exists at least a collection $S_i'$ of $\frac{\delta}{2(2c+1)}$ that are independently colourable locations by theorem 1. A deviation in any one of those locations passes undetected with probability less than 8/9 (as in theorem 3), and this probability is independent for each of these locations. This leads to a bound $\epsilon = p_{\text{fail}} \leqslant \left(\frac{8}{9}\right)^{\frac{\delta}{2(2c+1)}}$. The full proof is given in appendix E. □

Since the computation uses a fault-tolerant encoding, the number of qubits required scales accordingly. In particular, as in [4], there is an extra multiplicative cost $O(\log \frac{1}{\epsilon})$, where $\epsilon = \left(\frac{8}{9}\right)^d$, leading to overall complexity $O(\log \frac{1}{\epsilon}) \times O(N)$ similar to the classical output case.

## Acknowledgments

The authors would like to thank Vedran Dunjko, Alexandru Gheorghiu and Theodoros Kapourniotis for useful discussions. The authors are also grateful to Joe Fitzsimons for useful discussion regarding a robust fault tolerance scheme as proposed in appendix F. EK acknowledges funding through EPSRC grants EP/N003829/1 and EP/M013243/1.

## Appendix A. Proofs of dotted triple-graph construction

### *A.1. Proof of lemma 1*

**Proof.** First we note that after the break operations on red added vertices, all the vertices of different colours are disconnected. This follows since edges connecting different colour primary vertices were coloured by definition red, while all added vertices that were not red are connected with same colour vertices. Then, we need to show that the graph of each colour results in a graph identical to $D(G)$. To see this, note that for each vertex $v_i$ of the base graph, there is a white (black, green) vertex in $P_{v_i}$. Then for each edge $e(v_i, v_j)$ of the base graph $G$, there is a unique white (black, green) added vertex in $A_{e_{ij}}$ that joins the white vertex $p_w^{v_i} \in P_{v_i}$ and the white vertex $p_w^{v_j} \in P_{v_j}$ (and similarly for black and green). □

[8] It is important to note here, that $\delta/2$ is the number of different base-locations with non-trivial attacks, and not the number of qubits with non-trivial attacks.

### *A.2. Proof of lemma 2*

**Proof.** As the dotting operation only introduces vertices connected to vertices in $P(D(G))$, every vertex in $A(D(G))$ shares edges only with vertices in $P(D(G))$. Thus when the vertices in $P(D(G))$ and their associated edges are removed by the break operators, the vertices in $A(D(G))$ become disconnected. Similarly, since the dotting operation removes all edges between vertices in $P(D(G))$, hence every vertex in $P(D(G))$ shares edges only with vertices in $A(D(G))$. Thus when the vertices in $A(D(G))$ and their associated edges are removed by the break operators, the vertices in $P(D(G))$ become disconnected. □

### *A.3. Proof of lemma 3*

**Proof.** First we prove that a collection of base-locations satisfying this condition is ICL. From $\epsilon_i \cap \epsilon_j = \varnothing$ we can see that (i) for all primary base-locations in $\mathcal{E}$ no neighbouring base-location is in $\mathcal{E}$ and (ii) for each added base-location, the two neighbouring primary-locations $P_{v_i}, P_{v_j}$ are not in $\mathcal{E}$ and neither is any other added base-location set that has neighbours either of $P_{v_i}, P_{v_j}$. In other words, the sets of neighbours of added base-locations are disjoint. However, we already noted that a local-colouring of an added base-location is equivalent with a local-colouring of the two neighboring primary base-locations $P_{v_i}, P_{v_j}$. By replacing the local-colouring of added base-locations with that of the neighbouring primary base-locations, we reduce the local-colouring of the set $\mathcal{E}$ to that of a collection of local-colourings of primary base-locations. This is ICL since by the definition of trap-colouring no constraint is imposed between the colours of primary sets.
To prove the converse consider two locations $i$,$j$ such that $\epsilon_i \cap \epsilon_j \neq \varnothing$. Either of these is primary and the other is a neighbouring added base-location, or $i$ and $j$ are added base-locations sharing a common neighbour $k$. In the first case it is clear that the choice of colour at the primary set (say $i$) imposes constraints on the colours of the added base-location $j$. In the second case, the choise of colour at the added location $i$ can determine that of the neighbour location $k$ (for example a white added vertex that is connected with a primary vertex fixes the colour of that vertex to white). But then fixing the colours of the primary base-location $k$ in its turn imposes constraints for the other added neighbour $j$, and thus a local-colouring of $i$ and $j$ may not be consistent with a trap-colouring. □

## Appendix B. Proof of correctness (theorem 2)

**Proof.** To prove the correctness of the protocol we assume that the prover is honest and follows the instructions. This proof is very similar with [4]. We first consider the effect that the dummy qubits have. Dummies are equivalent with $Z$ measurement and therefore their effect is to break the graph at this particular vertex. In protocol 1 the dummies are placed at red vertices of a trap-colouring of the $DT(G)$ and on white added-vertices and black primary-vertices. According to lemmas 1 and 2 this results in having a copy of the dotted graph $(D(G))$ at the green vertices, and isolated qubits at the white primary vertices and black added vertices. Moreover, the quantum state that the isolated qubits are is $|+_{\theta_i}\rangle$.

The measurements on different (disconnected) graphs do not affect each other, so we consider separately the measurement pattern on the (green) dotted-graph $D(G)$ and the measurements in the isolated (trap) qubits.

The qubits in the computation graph $(D(G))$ are measured in the rotated basis $\delta_i = \phi'_i + \theta_i + \pi r_i$, while the graph is similarly rotated as each qubit (before the entangling

operations between the computation qubits) was in state $|+_{\theta_i}\rangle$. As in UBQC [26] this is identical with performing $\mathbb{M}_{\text{Comp}}$ to the non-rotated graph and results to the correct computation (by assumption), provided that the verifier, in order to account for the extra $\pi r_i$ rotation, sets $s_i = b_i \oplus r_i$ and uses $s_i$ to compute the next measurement angles.

The isolated traps (that are in state $|+_{\theta_i}\rangle$) are measured in $\delta_i = \theta_i + \pi r_i$ angle (as $\phi'_i = 0$ for dummy and trap qubits) and give deterministically the outcome $b_i = r_i$. This is precisely the outcome that the verifier needs to accept the computation as correct. Therefore, in the honest prover case, the verifier always accepts the output (traps correct) and as we saw in the previous paragraph, obtains as output the ideal (correct).

Finally, note that the dummy qubits are also measured. However, since they are disconnected from the rest of the qubits (they do not affect them), and their result contributes neither to the correct output nor to the accept/reject decision, the outcome of these measurements is irrelevant. □

## Appendix C. Proof of verification (theorem 3)

**Proof.** We now give some definitions taken from [4], before breaking the proof to five steps and exploring the places that we differ. The *output density operator* of the protocol is $B_j(\nu)$ and is given by

$$B_j(\nu) = \text{Tr}_B\left(\sum_b |b + c_r\rangle\langle b| C_{\nu_C,b}\, \Omega\, \mathcal{P}(\otimes^B |0\rangle\langle 0| \otimes |\Psi^{\nu,b}\rangle\langle\Psi^{\nu,b}|)\mathcal{P}^\dagger\, \Omega^\dagger\, C^\dagger_{\nu_C,b}|b\rangle\langle b + c_r|\right) \tag{C.1}$$

where we have the following definitions: The subscript $j$ of the operator $B$, corresponds to the strategy/deviation that the prover makes, and when $j = 0$ is the honest run where there is no deviation (and thus the operator $\Omega = \mathbb{I}$). The index $\nu$, collectively denotes all the random choices made by the verifier, i.e. $x, r, \theta, d$ and the positions of the traps $T$ (where the latter depends on the trap-colouring of the dotted triple-graph). When required, we make the further distinction between $\nu_T$ (parameters related with the trap) which are $r_t, \theta_t$ and the trap positions, and $\nu_C = \nu \setminus \nu_T$ (the remaining parameters). The $b$'s are the outcomes of the prover's measurement, $(c_r)_i = r_i$ for $i \notin T$ and $(c_r)_t = 0$ for $t \in T$, the subscript $B$ denotes tracing-out the prover's private registers. $C_{\nu_C,b}$ is the Pauli operator acting on the quantum output, that maps the final outcome to the correct one depending on the choices of random variables $\nu_C$ and the computation branch $b$. $\mathcal{P}$ is the unitary corresponding to implementing honestly the protocol. $\Omega$ is the deviation of the prover and is identity in the honest run. $|\Psi^{\nu,b}\rangle = |M^\nu\rangle \otimes_j |\delta^b_j\rangle$ is the initial state send by the verifier, that includes the quantum input and the $|+_\theta\rangle$ states which are jointly denoted as $|M^\nu\rangle$ and depend on the random choices, and the $|\delta\rangle$ registers correspond to the measurement angles (that depend on the branch of the computation $b$). Finally, $|\eta_t\rangle = |\theta_t\rangle$ if $t \in O$ while $|\eta_t\rangle = |r_t\rangle$ otherwise and the ideal state $|\Psi_{\text{ideal}}\rangle\langle\Psi_{\text{ideal}}| = \text{Tr}_{i\notin\{O\setminus\{T\}\}}(B_0(\nu))$ is the computation (green) output qubits when trivial deviation $\Omega = \mathbb{I}$ occurs.

For simplicity, in this proof, we have assumed that the initial state is pure and that the computation $\mathcal{P}$ to be performed is unitary and therefore the honest ideal state $|\Psi_{\text{ideal}}\rangle\langle\Psi_{\text{ideal}}|$ is also pure.

The probability of failure of the protocol is when the protocol returns 'accept' but the output is orthogonal to the honest ideal. This probability is given by

$$p_{\text{fail}} = \sum_\nu p(\nu)\text{Tr}(P^\nu_{\text{incorrect}} B_j(\nu)) \tag{C.2}$$

where

$$\begin{aligned} P^{\nu}_{\text{incorrect}} &= (\mathbb{I} - |\Psi_{\text{ideal}}\rangle\langle\Psi_{\text{ideal}}|) \otimes_{t\in T} |\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}| \\ &= P_{\perp} \otimes_{t\in T} |\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}| \end{aligned} \tag{C.3}$$

is the projection into the wrong subspace (orthogonal space to the correct ideal state) while it still remains within the accept subspace (where the traps succeed).

The proof has the following five steps. In **step 1** we express the attack using Kraus operators and Pauli matrices, in **step 2** we show that in order to lie in the incorrect subspace, at least one non-trivial attack to one qubit (of the dotted triple-graph) is required, and then we will replace the projection to the incorrect subspace with this restriction on the sum of allowed attacks. In **step 3** we will exploit the blindness of the prover to reduce the attack to Pauli attacks. In **step 4** we will show that the fewer the non-trivial attacks the greater the probability for the adversary, and thus we will restrict to the fewer allowed attacks (a single one). Finally, in **step 5** we will use a suitable partition of the qubits which will then leads to a constant bound for the $p_{\text{fail}}$.

**Step 1**: First we note that after tracing out the prover's register, the unitary $\Omega$ becomes a completely positive trace preserving map (CPTP), and can be expressed in terms of the Kraus operators $\{\chi_k\}$, where $\sum_k \chi_k \chi_k^{\dagger} = \mathbb{I}$. Moreover we express each Kraus operator as linear combination of Pauli operators $\chi_k = \sum_i \alpha_{ki}\sigma_i$ and $\sum_{k,i} \alpha_{ki}\alpha_{ki}^{*} = 1$. The matrix $\sigma_i$ is a tensor product of Pauli matrices, where if we want to specify the Pauli acting on qubit $\gamma$ we will denote it as $\sigma_{i|\gamma}$. We then get

$$\begin{aligned} p_{\text{fail}} &= \sum_{\nu} p(\nu)\text{Tr}(P^{\nu}_{\text{incorrect}} B_j(\nu)) \\ &= \sum_{b,i,j,k} \text{Tr}\left( \sum_{\nu} p(\nu)\alpha_{ki}\alpha_{kj}^{*} (P_{\perp} \otimes_{t\in T} |\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}|) |b + c_r\rangle\langle b| C_{\nu_C,b}\sigma_i \mathcal{P} |\Psi^{\nu,b}\rangle\langle\Psi^{\nu,b}| \mathcal{P}^{\dagger}\sigma_j C^{\dagger}_{\nu_C,b} |b\rangle\langle b + c_r| \right). \end{aligned} \tag{C.4}$$

**Step 2**: Again following [4], we can see that only terms that satisfy

$$\text{Tr}(P_{\perp}\sigma_i \mathcal{P} |\Psi^{\nu,b}\rangle\langle\Psi^{\nu,b}| \mathcal{P}^{\dagger}\sigma_j) \neq 0 \tag{C.5}$$

contribute to the $p_{\text{fail}}$. The terms that obey this are those necessarily within those that $|B_i| + |C_i| + |D_i^O| \geqslant 1$, which we will denote as $i \in E_i$ (and similarly $j \in E_j$), where the sets are defined as:

$$\begin{aligned} A_i &= \{\gamma \text{ s.t. } \sigma_{i|\gamma} = I \text{ and } \gamma \text{ qubit of the dotted triple-graph}\} \\ B_i &= \{\gamma \text{ s.t. } \sigma_{i|\gamma} = X \text{ and } \gamma \text{ qubit of the dotted triple-graph}\} \\ C_i &= \{\gamma \text{ s.t. } \sigma_{i|\gamma} = Y \text{ and } \gamma \text{ qubit of the dotted triple-graph}\} \\ D_i &= \{\gamma \text{ s.t. } \sigma_{i|\gamma} = Z \text{ and } \gamma \text{ qubit of the dotted triple-graph}\} \end{aligned} \tag{C.6}$$

and the superscript $O$ denotes subset of those sets that the $\gamma$ is output qubit. In other words, to corrupt the computation one either needs to flip the outcome of a measured qubit, or make any Pauli (other than the identity) if the attack is on the quantum output.

We have now imposed that the attacks $\sigma_i$ that contribute have at least one non-trivial Pauli attack at a qubit of the DT(G). This is not a sufficient condition to corrupt the computation in general (and send it to the $P_{\perp}$ subspace), but is a necessary condition.

To see this, we note that if we consider a $\sigma_i$ where $i \notin E_i$, then there is no choice of the secret parameters that would bring the state in the $P_\perp$ subspace. Here we take the worse-case scenario, where we assume that if there is some choice of secret parameters that a given attack could corrupt the computation, then we assume that it already is in the subspace $P_\perp$ and we only check what is the probability that this attack did not trigger any trap. For protocol 1 it is a single attack that could corrupt the computation. We then replace the projection on the $P_\perp$ subspace, with a restriction on the possible attacks, i.e. at the sum we only have terms corresponding to attacks that belong to the set $E_i$. Note, that if the computation was encoded in an fault-tolerant way (as is done in section 4.2), then the set $E_i$ requires greater number of non-trivial attacks. For now we take the more conservative view.

We then obtain the following expression:

$$p_{\text{fail}} \leqslant \sum_{k,b'} \sum_{\nu} p(\nu) \times \text{Tr}\Bigg( ( \otimes_{t\in T} |\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}| \otimes |b'\rangle\langle b'|) \left(\sum_{i\in E_i} \alpha_{ki}\sigma_i\right) \mathcal{P}|\Psi^{\nu,b'}\rangle\langle\Psi^{\nu,b'}|\mathcal{P}^\dagger \left(\sum_{i\in E_i} \alpha_{ki}\sigma_i\right)^\dagger \Bigg) \tag{C.7}$$

where $b' = \{b_i\}_{i\notin T}$ a substring of $b$ that excludes the value for the trap measurements (and we used that $\langle\eta_t^{\nu^T}||b_t\rangle = \delta_{\eta_t^{\nu_T},b_t}$).

**Step 3**: The next step is to exploit the fact that summing over the secret parameters of the verifier result to the prover being blind, and show that the only attacks that contribute are Pauli attacks, i.e. attacks that $\sigma_{i|\gamma} = \sigma_{j|\gamma}$ for all $\gamma$. Summing over $\nu_C$ we obtain

$$p_{\text{fail}} \leqslant \sum_{k,\nu_T} \sum_{i\in E_i} \sum_{j\in E_j} \alpha_{ik}\alpha^*_{jk} p(\nu_T) \times \text{Tr}\left( \otimes_{t\in T} |\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}| \sigma_i \left( \otimes_{t\in T} |\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}| \otimes_{t\in T} |\delta_t\rangle\langle\delta_t| \otimes \frac{I}{\text{Tr}(I)} \right) \sigma_j \right). \tag{C.8}$$

As all Pauli matrices but the identity are traceless, all terms in the sum are zero unless $\sigma_{i|\gamma} = \sigma_{j|\gamma}$ apart from the case that $\gamma \in T$. Then we use the fact that

$$\sum_{\theta_t,r_t} \text{Tr}(\langle\eta_t^{\nu_T}|\sigma_i|\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}|\sigma_j|\eta_t^{\nu_T}\rangle) = 0 \tag{C.9}$$

unless $\sigma_{i|t} = \sigma_{j|t}$ in the case that $t \in O$ and that for measured traps it suffices to sum over $r_t$, i.e. $\sum_{r_t} \text{Tr}(\langle\eta_t^{\nu_T}|\sigma_i|\eta_t^{\nu_T}\rangle\langle\eta_t^{\nu_T}|\sigma_j|\eta_t^{\nu_T}\rangle) = 0$ unless $\sigma_{i|t} = \sigma_{j|t}$. We then conclude that only terms that contribute are those that $\sigma_i = \sigma_j$. We thus obtain:

$$p_{\text{fail}} \leqslant \sum_{k} \sum_{i\in E_i} |\alpha_{ki}|^2 \sum_{T} p(T) \prod_{t\in T} \left( \sum_{\theta_t,r_t} p(\theta_t)p(r_t)(\langle\eta_t^{\nu_T}|\sigma_{i|t}|\eta_t^{\nu_T}\rangle)^2 \right) \tag{C.10}$$

where we broke the sum of $\nu_T$ to the choice of positions $T$, and the random choices of $\theta_t, r_t$, and we have taken the product of all those terms corresponding to the various white and black traps.

**Step 4**: In this step, we will prove that to maximise the value of the bound of the probability of $p_{\text{fail}}$, the best strategy is to do the least number of attacks allowed by the constraint obtained at step 2, which in our case, is a single attack. Then at the next step we will bound this maximum value. We have

$$p_{\text{fail}} \leqslant \sum_k \sum_{i \in E_i} |\alpha_{ki}|^2 f(i) \tag{C.11}$$

where $f(i) := \sum_T p(T) \prod_{t \in T} \left( \sum_{\theta_t, r_t} p(\theta_t) p(r_t) (\langle \eta_t^{\nu_T} | \sigma_{i|_t} | \eta_t^{\nu_T} \rangle)^2 \right)$. From $\sum_{ik} |\alpha_{ik}|^2 = 1$ we conclude that $p_{\text{fail}}$ is maximised when $|a_{ik}| = 0$ for all $i \notin E_i$. Then we have a convex combination of values $f(i)$. Let $f(m) = \max_{i \in E_i} f(i)$, then and it follows that if this is maximum for the single value $m$, then by choosing $|a_{ik}| = 0$ for all $i \neq m$ the bound for $p_{\text{fail}}$ is maximised.

$$\begin{aligned} p_{\text{fail}} &\leqslant f(m) = \max_{i \in E_i} f(i) \\ &\leqslant \max_{i \in E_i} \sum_T p(T) \prod_{t \in T} \left( \sum_{\theta_t, r_t} p(\theta_t) p(r_t) (\langle \eta_t^{\nu_T} | \sigma_{i|_t} | \eta_t^{\nu_T} \rangle)^2 \right). \end{aligned} \tag{C.12}$$

In other words, we obtain a bound by considering a single $\sigma_i$ that belongs to the set $E_i$ and maximises the expression we have. The following expression involves a product of positive numbers, that are all less or equal to unity:

$$\prod_{t \in T} \left( \sum_{\theta_t, r_t} p(\theta_t) p(r_t) (\langle \eta_t^{\nu_T} | \sigma_{i|_t} | \eta_t^{\nu_T} \rangle)^2 \right). \tag{C.13}$$

In particular we can see that the terms in the product of equation (C.13) are unity for all trap positions that $\sigma_{i|_t}$ is trivial, i.e. $\sigma_{i|_k} \notin \{X, Y\}$ if $k$ is not output, or $\sigma_{i|_k} \notin \{X, Y, Z\}$ if $k$ is an output qubit. It is clear that this expression is bigger the more terms contain trivial attacks on traps. In other words, if we have two possible attacks $\sigma_i$ and $\sigma_{i'}$, where for all $\gamma$ that $\sigma_{i|_\gamma}$ is non-trivial it is equal to $\sigma_{i'|_\gamma}$ (but there are $\gamma$ that $\sigma_{i'|_\gamma}$ is non-trivial while $\sigma_{i|_\gamma}$ is trivial), then $f(i) \geqslant f(i')$. Therefore the term that maximises $p_{\text{fail}}$ corresponds to an attack $\sigma_i$ that has the fewest (possible, i.e. compatible with $E_i$) non-trivial terms.
From step 2 we obtained that the set $E_i$ has at least one non-trivial Pauli attack, so it follows that the bound of the $p_{\text{fail}}$ we compute is maximised when there is exactly one non-trivial Pauli attack. It is important to note however, that the set $E_i$ will be different in section 4.2 where we consider fault-tolerant encoding of the computation and the corresponding $\sigma_i$ will involve greater number of non-trivial attacks. In that case, the set of attacks that can possibly corrupt the computation (and thus send it to $P_\perp$ subspace) changes (i.e. $E_i$ differs).

**Step 5**: We will now use the partition of the qubits of the dotted triple-graph, to the subsets $P_v, A_e$ corresponding to vertices and edges of the base graph. The way that this partition is chosen does not reveal any new information to the prover and does not depend on the choice of trap-colouring, i.e. on the positions of the traps.
We have established that the optimal strategy for the prover in order to maximise the value of the bound for the $p_{\text{fail}}$ we compute, is to make a single non-trivial attack at one qubit of the dotted triple-graph. Let us assume that this single position is $\beta$ and we know that it belongs to either a set $P_{v_\beta}$ or a set $A_{e_\beta}$ depending on whether the non-trivial attack is done on a qubit belonging to a primary set $P_{v_\beta}$ or an added set $A_{e_\beta}$. When it is not clear if the set is primary or added, we will use $F_\beta$ which simply means that $F_\beta = P_{v_\beta}$ if $\beta$ is at a primary location and $F_\beta = A_{e_\beta}$ if $\beta$ is at an added location.

We then break the $p(T)$ which is the probability of different trap configurations, using the structure of the subsets $P_v, A_e$, i.e. $p(T) = p(t_1 \in P_{v_1}, t_2 \in P_{v_2}, \cdots, t_k \in A_{e_1}, \cdots)$. Therefore, given a

single attack at set $F_\beta$, we can sum over all the other sets (all the other positions do not appear in the remaining expression) and obtain $\sum_T p(T) = \sum_{t_\beta \in F_\beta} \sum_{t \notin F_\beta} p(T) = \sum_{t_\beta \in F_\beta} p(t_\beta)$. We obtain

$$p_{\text{fail}} \leqslant \max_{i \in E_i} \sum_{t_\beta \in F_\beta} \sum_{\theta_{t_\beta}, r_{t_\beta}} p(t_\beta) p(\theta_{t_\beta}) p(r_{t_\beta}) (\langle \eta^\nu_{t_\beta} | \sigma_{i|t_\beta} | \eta^\nu_{t_\beta} \rangle)^2. \tag{C.14}$$

It is important to note that $\sigma_{i|t_\beta}$ is the identity (or $Z$ for qubits not in the output) if $\beta \neq t_\beta$ while it is non-trivial otherwise, therefore the $(|F_\beta| - 1)$ terms of the sum will be the unity, while one term will be less than one[9]. The above expression depends on whether the set $F_\beta$ is the output set, or in the case that is a measured set on whether it is a primary or added set. It will be the prover that chooses which is the set of the attack, and thus the bound will be the highest of these values. We consider separately each case. We define the quantity

$$g(i, F_\beta) = \sum_{t_\beta \in F_{t_\beta}} \sum_{\theta_{t_\beta}, r_{t_\beta}} p(t_\beta) p(\theta_{t_\beta}) p(r_{t_\beta}) (\langle \eta^\nu_{t_\beta} | \sigma_{i|t_\beta} | \eta^\nu_{t_\beta} \rangle)^2 \tag{C.15}$$

where the function $g$ has an explicit dependence on which set $F_\beta$ the non-trivial attack belongs to. In particular, we will denote $P^O_{v_\beta}$ if the non-trivial attack is on an output set (note that output qubits are only primary), and $P_{v_{\beta'}}$ if it is on a measured primary set and $A_{e_{\beta'}}$ if it is on a measured added set. We will separately compute the maximum of $g(i, P^O_{v_\beta}), g(i, P_{v_{\beta'}}), g(i, A_{e_{\beta'}})$ for $i \in E_i$ and the bound will be the maximum of those three.

We start with the output qubits

$$\begin{aligned} g(i, P^O_{v_\beta}) &= \frac{1}{16|P^O_{v_\beta}|} \sum_{t_\beta \in P^O_{v_\beta}} \sum_{\theta_t, r_t} (\langle +_\theta | \sigma_{i|t_\beta} | +_\theta \rangle)^2 \\ &= \frac{1}{16 \times 3} \sum_{\theta_t, r_t} \Big( 1 \cdot (|P^O_{v_\beta}| - 1) + 1 \cdot (\langle +_\theta | \sigma_{i|t_\beta} | +_\theta \rangle)^2 \Big) \\ &\leqslant \frac{1}{16 \times 3} \Big( 16 \cdot (|P^O_{v_\beta}| - 1) + 8 \Big) \\ &\leqslant \left( 1 - \frac{1}{2|P^O_{v_\beta}|} \right) = \frac{5}{6} \end{aligned} \tag{C.16}$$

where we used that $\sum_\theta (\langle +_\theta | \sigma | +_\theta \rangle)^2 \leqslant 4$ for $\sigma \neq \mathbb{I}$ and that $|P^O_{v_\beta}| = 3$ since output qubits are primary.

Now we consider the measured qubits similarly (using $F_{\beta'}$ to denote either the primary or added measured set), to obtain

$$\begin{aligned} g(i, F_{\beta'}) &= \frac{1}{16|F_{\beta'}|} \sum_{t_\beta \in F_{\beta'}} \sum_{\theta_t, r_t} (\langle r_{t_\beta} | \sigma_{i|t_\beta} | r_{t_\beta} \rangle)^2 \\ &= \frac{1}{16|F_{\beta'}|} \sum_{r_t} \Big( 8 \cdot (|F_{\beta'}| - 1) + 8 \cdot (\langle r_{t_\beta} | \sigma_{i|t_\beta} | r_{t_\beta} \rangle)^2 \Big) \\ &= \frac{1}{16|F_{\beta'}|} \Big( 16 \cdot (|F_{\beta'}| - 1) \Big) \\ &= \left( 1 - \frac{1}{|F_{\beta'}|} \right) \leqslant \frac{8}{9} \end{aligned} \tag{C.17}$$

[9] It turns out that the not-unity term, is zero for measured qubits, while it can be up to 1/2 for output qubits.

where the last step the equality holds if the attack is on added qubits where $|F_{\beta'}| = |A_{e_{\beta'}}| = 9$. For primary measured sets $P_{v_{\beta'}}$ the bound is only (2/3) and thus is lower. It follows that the overall bound we obtain (worst-case) is

$$p_{\text{fail}} \leqslant \left(\frac{8}{9}\right). \tag{C.18}$$

Since this bound is obtained when the attack is on a measured (added) qubit, the bound is the same when the output is fully classical. □

## Appendix D. Protocol for classical output

**Protocol 4.** Boosted Verifiable UBQC using dotted triple-graph for classical output.

- Verifier chooses computation with classical, deterministic output (e.g. decision problem).
- Verifier chooses a number $d$, where $d = \frac{\log \epsilon}{\log(8/9)}$ and the desired security level is $\epsilon$.
- For each $i \in \{1, \cdots, d\}$ **Verifier follows the steps of the protocol 1 with random different choices of secret parameters.** If the verifier accepts the computation, they register the classical output as $O_i$ and store it.
- If the verifier rejected at any single repetition of protocol 1, they reject the overall computation. If not, they compare the classical outputs $O_i$ and if all of them are identical, they accept this output as the output of the computation.

**Theorem 5 (Verification).** *Protocol 4 is* $\left(\frac{8}{9}\right)^d$*-verifiable where the output is classical and d is the number of repetitions.*

**Proof.** We have multiple repetitions and if all of them return the same output $O$, then the probability that this is not the correct output is bounded by the probability that *all* repetitions failed (and resulted to the same deviation). Since the different repetitions have the same outcome it means that if a single of those repetitions is successful then the output $O$ is the correct output. From theorem 3 we know that the probability that a single repetition fails is 8/9. Then the probability that all the $d$ repetitions fail is $\left(\frac{8}{9}\right)^d$. □

In the case of classical output, there is an alternative construction to the dotted triple-graph that could decrease further the (linear) overhead. In particular, instead of having the dotted triple-graph $DT(G)$ one could consider three copies of the dotted base graph $D(G)$. We will name this the *three dotted copies* construction. The one copy will be used for computation, while the other two for white traps (on primary vertices) and black traps (on added vertices). This construction is global, in the sense that the decision of which vertices are in which graph is made from the beginning and cannot be decided independently per base graph vertex $v_i$. It follows that the location of the traps is totally correlated globally and there is no way to amplify the success probability in the quantum output case. This is the reason we focused on the dotted triple-graph construction for the quantum output case. For the classical output however, the three dotted copies construction works.

**Theorem 6 (Three dotted-copies verification).** *Protocol 5 is* $\left(\frac{2}{3}\right)^d$*-verifiable where the output is classical and d is the number of repetitions.*

**Protocol 5.** Boosted three dotted copies Verifiable UBQC for classical output.

- Verifier chooses a computation that has classical and deterministic output (e.g. a decision problem).
- Verifier chooses a number $d$, where $d = \frac{\log \epsilon}{\log(2/3)}$ and the desired security level is $\epsilon$.
- For each $i \in \{1, \cdots, d\}$ **Verifier follows steps of protocol 1 using three dotted-copies instead of dotted triple-graph and with random different choices of secret parameters for every run.** If the verifier accepts the computation, they register the classical output as $O_i$ and store it.
- If the verifier rejected at any single repetition of the modified protocol 1, they reject the overall computation. If not, they compare the classical outputs $O_i$ and if all of them are identical, they accept this output as the output of the computation.

**Proof.** Following the proof of theorem 3, at step 2 in order to corrupt the computation the prover needs to make at least one non-trivial attack. However, the prover is blind about which of the three graphs is the computation, white and black trap graph. Therefore, it has probability 1/3 that the attack coincides with a trap graph of the same type of the attack location (i.e. if it attacks a primary vertex, then with probability 1/3 the attack was on a qubit that belongs in the white graph, while if the attack was on an added vertex with probability 1/3 the attack was on a qubit that belongs in the black graph). For classical output the non-trivial attacks are $\{X,Y\}$ only (all qubits are measured), and thus the attack is deterministically detected when it hits a trap, as $\langle \eta_t^{\nu} | \sigma_{X/Y} | \eta_t^{\nu} \rangle = \langle r_t | \sigma_{X/Y} | r_t \rangle = 0$. Therefore the probability of failure is $p_{\text{fail}} \leqslant 2/3$.

To amplify this probability, we can simply repeat the protocol $d$ times, and if all classical outputs agree in all the runs then the probability that the computation was corrupted is bounded by $p_{\text{fail}} \leqslant \left(\frac{2}{3}\right)^d$. Moreover, the number of qubits required per repetition is $3|V(G)| + 3|E(G)| \leqslant 3(1+c)|V(G)$. Both in terms of failure probability and in terms of the (linear in both cases) number of qubits per repetition, the three copies construction gives better result. □

## Appendix E. Proof of verification for quantum output (theorem 4)

**Proof.** We assume that there is a fault-tolerant encoding of the computation, that when done in MBQC, corrects or detects all errors that have fewer than $\delta$ number of errors when the computation is performed on the base graph $G$. Any operation on a measured qubit, diagonal in the computational basis ($\sigma_i \in \{I, Z\}$) does not alter the computation. Therefore errors that can contribute to corrupting a single logical qubit, involve errors $\sigma_i \in \{X, Y\}$ for measured qubits and $\sigma_i \in \{X, Y, Z\}$ for output qubits.

Considering the dotted base graph $D(G)$, one can easily see that any (non-trivial) error on an added qubit $a_{e_{ij}}$, is equivalent with a local error on each of the two primary qubits that are neighbours $p_{v_i}, p_{v_j}$ (see also [4]). If to corrupt a computation one needs $\delta$ errors on primary qubits of the base graph $G$, it follows that to corrupt the computation when done on the dotted base graph $D(G)$ one needs at least $\delta/2$ errors on qubits of the dotted base graph $D(G)$.

We now turn back to step 2 of the proof of theorem 3 and we see that the set $E_i$ of attacks that could possibly corrupt the computation, should include non-trivial attack in at least $\delta/2$ *different* sets $P_v, A_e$ (which collectively we call $F_\beta$). It is important to note that within each of the sets $F_\beta$ there is a single computation-graph qubit and therefore not only the prover needs to perform $\delta/2$ non-trivial attacks, but they should also be done on at least $\delta/2$ different location sets. The prover of course could choose to attack multiple qubits of the same set $F_\beta$, but in order to hit at least $\delta/2$ computation qubits, the sets that they perform non-trivial attacks should also be at least $\delta/2$.

Any given attack $\sigma_i$ is characterised by the set $S_i$ of locations on the dotted base graph $D(G)$, that it has at least one non-trivial attack, which in the case $\sigma_i \in E_i$ it means that $|S_i| \geqslant \delta/2$ .

Following step 3 and 4 of the proof of theorem 3, we reach equation (C.12). From this expression we can again see that the fewer the positions of non-trivial attack (consistent with $E_i$), the greatest the value of this bound is. We already know that we need at least $\delta/2$ sets $F_\beta$ with non-trivial attacks, so it follows that the maximum is achieved when there are exactly $\delta/2$ different sets $F_\beta$ with exactly a single attack in each.

To proceed further we need to decompose the probability of different configuration of traps $p(T)$ to this of individual sets. This is not in general possible since there are correlations between the traps of (neighbouring) sets. To this point we should note that fixing a configuration of traps is identical with giving a trap-colouring as in definition 3.

From theorem 1, we know that given a collection $S_i$ of $\delta/2$ locations on the dotted base graph $D(G)$ there are at least a collection $S_i'$ of $\frac{\delta}{2(2c+1)}$ that are independently colourable locations, i.e. $|S_i'| = \lceil \frac{\delta}{2(2c+1)} \rceil$. To obtain an upper bound on the failure probability, we set $\sigma_{i|\gamma} = I$ for all $\gamma$ that belong to locations in $S_i \setminus S_i'$. This change is non-decreasing for the expression for the bound given in equation (C.12). Now the only locations that have non-trivial attacks are those in $S_i'$, and we have

$$p_{\text{fail}} \leqslant \max_{i \in E_i} \prod_{\beta=1}^{|S_i'|} \sum_{t_\beta \in F_\beta} p(t_\beta) \sum_{\theta_{t_\beta}, r_{t_\beta}} p(\theta_{t_\beta}) p(r_{t_\beta}) (\langle \eta_{t_\beta}^{\nu} | \sigma_{i|t_\beta} | \eta_{t_\beta}^{\nu} \rangle)^2 \tag{E.1}$$

where we used the fact that $p(T) = p(t_1 \in P_{v_1}, t_2 \in P_{v_2}, \cdots, t_k \in A_{e_1}, \cdots)$ and for a collection $S_i'$ of independent colourable locations,

$$\sum_{t_\beta \in S_i'} \left( \sum_{t_\beta \notin S_i'} p(T) \right) = \sum_{t_\beta \in S_i'} \left( \prod_{\beta=1}^{|S_i'|} p(t_\beta) \right). \tag{E.2}$$

We can see this due to the fact that after summing all locations apart from those in $S'$, the probability for choosing the location of the trap within each set is independent and thus the joint probability is simply their product. Now, each term in the product of equation (E.1) is bounded by 8/9 as proven in the previous section and therefore we obtain

$$p_{\text{fail}} \leqslant \left( \frac{8}{9} \right)^d \tag{E.3}$$

where we define $d = |S_i'| = \lceil \frac{\delta}{2(2c+1)} \rceil$ □

As a final remark, we should note that the value $d = \lceil \frac{\delta}{2(2c+1)} \rceil$ is the minimum number of ICL that exist and in particular cases this number can be greater and thus the probability of success of the verification protocol also becomes greater for those cases.

## Appendix F. Consequences for existing verification protocols

The dotted triple-graph construction can be used to improve a number of existing verification protocols and here we indicate three of them. First we consider the specific case where the computation is done using the Raussedorf–Harrington–Goyal (RHG) [29] encoding and the related graph is $\mathcal{G}_{\mathcal{L}}$. Following our construction instead of the dotted-complete graph of [4],

we obtain the dotted triple-RHG graph $DT\mathcal{G}_{\mathcal{L}}$. This graph state has linear number of qubits (as the maximum degree of the graph is 4). With the same choices of parameters as in [4], it can detect or corrects any deviation that has fewer than $\delta/2$ errors. From our results in the previous section, it follows that we obtain a linear-complexity verification protocol with exponential security bound given by $p_{\text{fail}} \leqslant \left(\frac{8}{9}\right)^{\lceil \frac{\delta}{18} \rceil}$.

The second application is that it can be used to improve verifiable fault-tolerant protocols. Assuming that there are errors due to noise (non-adversarial), the protocols given earlier in the text and in other VBQC protocols could face a problem. In particular, honest errors due to noise could make trap measurement fail and lead us to reject the output even in honest runs where the computation is not corrupted. Here we should stress that both in this paper and in [4] the use of fault-tolerant encoding was in order to amplify the security and *not* to correct the computation from errors caused by honest noise. However, one *can* construct a fault-tolerant verification protocol, at least for classical output, and one such example is presented in [13]. The starting graph used to obtain the fault-tolerant protocol of [13] was the brickwork graph that has a single trap. Then a fault-tolerant encoding was done followed by the repetition technique used to amplify the success probability. However the number of repetitions to maintain a constant level of security increased with the size of the computation. By using the dotted triple-brickwork instead, as the first step of the construction in [13], we can achieve exponential security with a constant number of repetitions. This would essentially bring down the number of qubits required from $O(n^2)$ to $O(n)$.

The third application is that we can directly use the dotted triple-graph construction for the verifiable measurement-only protocols [9, 10]. In particular [9] is essentially the online version of [4], and the technique to include traps in the graph is equivalent. It follows that if the resource used instead of a dotted-complete graph is a dotted triple-RHG graph then the number of qubits required will reduce from quadratic to linear. In [10] the verifier, instead of including traps, uses $2k + 1$ copies of a universal graph. In order to test the honesty of the prover it makes stabiliser measurements to $2k$ copies of the desired graph while it performs the computation on the final copy. However, there is always at least a $1/(2k + 1)$ probability that the computation is corrupted and not detected (e.g. one picks one copy and attacks all the qubits of that copy). Using the dotted triple-graph construction, modified for the measurement-only protocols, this probability can be made exponentially small while still using only linear number of qubits. In other words in [10] a malicious prover can choose one copy and corrupt all its qubits without diminishing their chances compared to their chances when corrupting a single qubit, as the positions of traps are correlated, i.e. a copy is either fully a test copy or a computation copy. On the other hand, in our local construction, for each base-location, the choice of computation and test qubits is independent.

## References

[1] Bremner M, Jozsa R and Shepherd D 2011 Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy *Proc. R. Soc.* A **467** 459
[2] Aaronson S and Arkhipov A 2011 The computational complexity of linear optics *Proc. of the 43rd Annual ACM Symp. on Theory of Computing* p 333
[3] Aharonov D, Jones V and Landau Z 2006 A polynomial quantum algorithm for approximating the Jones polynomial *Proc. of the 38th Annual ACM Symp. on Theory of Computing* p 427
[4] Fitzsimons J F and Kashefi E 2012 Unconditionally verifiable blind computation (arXiv:1203.5217)
[5] Aharonov D, Ben-Or M and Eban E 2010 Interactive proofs for quantum computations *Proc. of Innovations in Computer Science* p 453
[6] Barz S, Fitzsimons J F, Kashefi E and Walther P 2013 Experimental verification of quantum computation *Nat. Phys.* **9** 727–31

[7] Kapourniotis T, Dunjko V and Kashefi E 2015 On optimising quantum communication in verifiable quantum computing 2015 *Proc. of the 15th Asian Quantum Information Science Conf.* pp 23–5
[8] Kapourniotis T, Kashefi E and Datta A 2014 Blindness and verification of quantum computation with one pure qubit *9th Conf. on the Theory of Quantum Computation, Communication and Cryptography*
[9] Morimae T 2014 Verification for measurement-only blind quantum computing *Phys. Rev.* A **89** 060302
[10] Hayashi M and Morimae T 2015 Verifiable measurement-only blind quantum computing with stabilizer testing *Phys. Rev. Lett.* **115** 220502
[11] Reichardt B W, Unger R F and Vazirani U 2013 Classical command of quantum systems *Nature* **496** 456–60
[12] McKague M 2016 Interactive proofs for BQP via self-tested graph states *Theory Comput.* **12** 1–42
[13] Gheorghiu A, Kashefi E and Wallden P 2015 Robustness and device independence of verifiable blind quantum computing *New J. Phys.* **17** 083040
[14] Hajdušek M, Pérez-Delgado C A and Fitzsimons J F 2015 Device-independent verifiable blind quantum computation (arXiv:1502.02563)
[15] Dunjko V, Fitzsimons J F, Portmann C and Renner R 2014 Composable security of delegated quantum computation *Advances in Cryptology (Lecture Notes in Computer Science* vol 8874) ed P Sarkar and T Iwata (Berlin: Springer) pp 406–25
[16] Broadbent A 2015 How to verify a quantum computation (arXiv:1509.09180)
[17] Pappa A, Chailloux A, Wehner S, Diamanti E and Kerenidis I 2012 Multipartite entanglement verification resistant against dishonest parties *Phys. Rev. Lett.* **108** 260502
[18] Markham D and Marin A 2015 Practical sharing of quantum secrets over untrusted channels *Information Theoretic Security* (*Lecture Notes in Computer Science* vol 9063) ed A Lehmann and S Wolf (New York: Springer) pp 1–14
[19] Bell B A, Markham D, Herrera-Mart D A, Marin A, Wadsworth W J, Rarity J G and Tame November M S 2014 Experimental demonstration of graph-state quantum secret sharing *Nat. Commun.* **5** 5480
[20] Broadbent A, Gutoski G and Stebila D 2013 Quantum one-time programs *Advances in Cryptology CRYPTO 2013* (*Lecture Notes in Computer Science* vol 8043) ed R Canetti and J A Garay (Berlin: Springer) pp 344–60
[21] Aliferis P and Leung D W 2004 Computation by measurements: a unifying picture *Phys. Rev.* A **70** 062314
[22] Danos V, Kashefi E and Panangaden P 2007 The measurement calculus *J. ACM* **548**
[23] Raussendorf R and Briegel H J 2001 A one-way quantum computer *Phys. Rev. Lett.* **86** 5188–91
[24] Childs A M, Leung D W and Nielsen M A 2005 Unified derivations of measurement-based schemes for quantum computation *Phys. Rev.* A **71** 032318
[25] Hein M, Eisert J and Briegel H J 2004 Multiparty entanglement in graph states *Phys. Rev.* A **69** 062311
[26] Broadbent A, Fitzsimons J and Kashefi E 2009 Universal blind quantum computation *Proc. of the 50th Annual Symp. on Foundations of Computer Science* (IEEE Computer Society) pp 517–26
[27] Kashefi E and Wallden P 2016 Extending the delegated verifiable blind quantum computation functionality (arXiv:1606.06931)
[28] Networked quantum information technologies hub. (www.nqit.ox.ac.uk)
[29] Raussendorf R, Harrington J and Goyal K 2007 Topological fault-tolerance in cluster state quantum computation *New J. Phys.* **9** 199