QFactory: classically-instructed remote secret qubits preparation

Alexandru Cojocaru[1*], Léo Colisson[2†], Elham Kashefi[1,2‡], Petros Wallden[1§]

[1] School of Informatics, University of Edinburgh,
10 Crichton Street, Edinburgh EH8 9AB, UK
[2] Laboratoire d'Informatique de Paris 6 (LIP6), Sorbonne Université,
4 Place Jussieu 75252 Paris CEDEX 05, France

Abstract. The functionality of classically-instructed remotely prepared random secret qubits was introduced in (Cojocaru et al 2018) as a way to enable classical parties to participate in secure quantum computation and communications protocols. The idea is that a classical party (client) instructs a quantum party (server) to generate a qubit to the server's side that is random, unknown to the server but known to the client. Such task is only possible under computational assumptions. In this contribution we define a simpler (basic) primitive consisting of only BB84 states, and give a protocol that realizes this primitive and that is secure against the strongest possible adversary (an arbitrarily deviating malicious server). The specific functions used, were constructed based on known trapdoor one-way functions, resulting to the security of our basic primitive being reduced to the hardness of the Learning With Errors problem. We then give a number of extensions, building on this basic module: extension to larger set of states (that includes non-Clifford states); proper consideration of the abort case; and verifiablity on the module level. The latter is based on "*blind self-testing*", a notion we introduced, proved in a limited setting and conjectured its validity for the most general case.

Keywords: Classical delegated quantum computation · Learning With Errors · Provable security

* a.d.cojocaru@sms.ed.ac.uk
† leo.colisson@ens-paris-saclay.fr
‡ ekashefi@inf.ed.ac.uk
§ petros.wallden@ed.ac.uk

Table of Contents

1 Introduction

In the coming decades, advances in quantum technologies may cause major shifts in the mainstream computing landscape. In the meantime, we can expect to see quantum devices with high variability in terms of architectures and capacities, the so-called noisy, intermediate-scale quantum (NISQ) devices [Pre18] (such as those being developed by IBM, Rigetti, Google, IonQ) that are currently available to users via classical cloud platforms. In order to be able to proceed to the next milestone for the utility of these devices in a wider industrial base, the issues of privacy and integrity of the data manipulation must be addressed.

Early proposals for secure and verifiable delegated quantum computing based on simple obfuscation of data already exist [AS03, Chi05, ABEM17, BFK09, DKL11, MF12, MPDF13, GMMR13, MDK15, FK17]. However, these schemes require a reliable long-distance quantum communication network, connecting all the interested parties, which remains a challenging task.

For these reasons, there has recently been extensive research focusing on the practicality aspect of secure and verifiable delegated quantum computation. One direction is to reduce the required communications by exploiting classical fully-homomorphic-encryption schemes [BJ15, DSS16, ADSS17], or by defining their direct quantum analogues [Lia15, OTF15, TKO$^+$16, LC17]. Different encodings, on the client side, could also reduce the quantum communication [MPDF13, GMMR13]. However, in all these approaches, the client still requires some quantum capabilities. While no-go results indicate restrictions on which of the above properties are jointly achievable for classical clients [AGKP14, YPDF14, ACGK17, NS17], recent breakthroughs based on post-quantum secure trapdoor one-way functions, paved the way for developing entirely new approaches towards fully-classical client protocols for emerging quantum servers. The first such procedures were proposed in [Mah18a] allowing a classical client to securely delegate a universal quantum computation to a remote untrusted server. The key technical idea was the ability to perform a CNOT quantum gate that is controlled by an encrypted classical bit. To achieve this a classical primitive of trapdoor claw-free functions pair was used. It was later followed by the work of [Bra18], where the construction achieved stronger security guarantee and based on more standard cryptographic assumptions. Building on this, a single device certifiable randomness was achieved in [BCM$^+$18], where the randomness is information theoretical, but the certification is based on the computational limitations of quantum devices. Finally, using post-hoc verification of quantum computations [FHM18], and the above ideas to generate a single qubit "blind measurement device", [Mah18b] gave the first protocol achieving classical client verification of universal quantum computation.

All these constructions, while they used similar techniques, proved the desired properties in a monolithic way. An alternative approach was taken in [CCKW18] where the idea was to replace the quantum channel (that is used in many different protocol implementing blind and/or verifiable quantum computation) with a module running between a classical client and a quantum server. It was shown then how a classical client could use this module (referred to as QFactory) to

achieve secure delegated universal quantum computing, but potentially also, other functionalities such as multi-party quantum computation. However, the security proof was made in a weak "honest-but-curious" model, and the full proof of security was left as an open question. In this paper, we extend the security proof to a fully malicious adversary. All our proofs are made using reductions to hardness assumptions (namely LWE), and the simplicity of the main protocol suggests that an extention to a composable model such as Universal Composability [Can00] or Abstract Cryptography (AC) [MR11] should be possible (but left as a future work).

Concurrently with our work, [GV19] also took this modular approach. Technically they followed closely the ideas from [Mah18a, Mah18b, BCM+18], but the basic primitive they derive (in a verifiable version) is the one we introduced in [CCKW18]. They gave protocols for a *verifiable* version of the secret single qubit generation, while they also gave a proof in the AC model. Their AC proof relies on a strong hypothesis that they call "measurement buffer" that forces the adversary to give the state that he is supposed to measure to the simulator, enforcing (essentially) a trusted measurement. They also state that the stand-alone proof does not require this assumption. To our view, this assumption is very high price for moving to the AC framework, which is why in our current work we do not focus on composability while we work towards resolving this issue as part of the future work we mentioned. Moreover, in [GV19] they do not investigate a crucial "abort" case of the protocol, which is related to the properties of the functions required for the protocol implementation. Specifically, properties such as two-regularity can only be achieved probabilistically, causing the security of the protocol to fail whenever the function property is not satisfied. This means that they need to use a familly of functions that is secure only under the assumption that SIVP_γ is hard for a superpolynomial γ, while the standard assumption is that SIVP_γ is secure only for a polynomial γ ([Bra18]).

Following the modularity of [CCKW18], we present a universal yet minimal functionality module that is fully secure and verifiable at the module level and could be used as a black box in other client-server applications to replace the need for a reliable long-distance quantum communication network. The price one has to pay is a reduction from information-theoretic security (achievable using quantum communication) to post-quantum computational security via our modules. The ultimate vision would be to develop a hybrid network of classical and quantum communication channels, depending on the desired security level and the technology development of NISQ devices allowing classical or quantum links [WEH18].

1.1 Our Contributions

In [CCKW18] was defined a classical client - quantum server functionality of delegated pseudo-secret random qubit generator (PSRQG) that can replace the need for quantum channel between parties in certain quantum communication protocols, with the only trade-off being that the protocols would become compu-

tationally secure (against *quantum* adversaries). However, the proof of security was done in a weak model called "honest-but-curious". In this paper:

1. We present a new protocol called Malicious 4-states QFactory in Section 3 that achieves the functionality of classically instructed remote secret generation of the states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ (known as the BB84 states), given 2 cryptographic functions: 1) a trapdoor one-way function that is quantum-safe, two-regular and collision resistant and 2) a homomorphic, hardcore predicate. The novelty of this new protocol reflects in both simplicity of construction and proof, as well as enhanced security, namely the protocol is secure against any arbitrarily deviating adversary. The target output qubit set is one of the four BB84 states, states that form the core requirement of any quantum communication protocol.
   Then, in Subsection 3.3, we present the security of the Malicious 4-states QFactory against any fully malicious server, by proving that the basis of the generated qubits are completely hidden from any adversary, using the properties of the two functions, the security being based on the hardness of the Learning with Errors problem.
2. While the above-mentioned results do not depend on the specific function used, the existence of such functions (with all desired properties) makes the functionality a practical primitive that can be employed as described in this paper. In Section 4, we describe how to construct the two-regular, collision resistant, trapdoor one-way family of functions and the homomorphic, hardcore predicate. Furthermore, we prove using reductions in Subsection 4.2 that the resulting functions maintain all the required properties.
3. In order to demonstrate the modular construction of the basic Malicious 4-states QFactory, we also present in Section 5, a secure and efficient extension to the functionality of generating 8 states, called the Malicious 8-states QFactory protocol (where the security refers to the fact that the basis of the new state is completely hidden). The set of output states $\left\{|+_\theta\rangle \mid \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\right\}$ (no longer within the Clifford group) are used in various protocols, including protocols for verifiable blind quantum computation.
4. While the protocol introduced in Section 3 requires (for the security proof) a family of functions having 2 preimages with probability super-polynomially close to 1, we also define in Section 7 a protocol named Malicious-Abort 4-states QFactory, that is secure when the functions have 2 preimages with only a constant (greater than 1/2) probability. Indeed, even if the parameters used for the first category of functions are implicitly used in some protocols [Mah18a], the second category of functions is strictly more secure and more standard in the cryptographic literature [Bra18]. The Malicious-Abort 4-states QFactory protocol is proven secure also for this second category of functions, assuming that the classical Yao's XOR lemma also applies for one-round protocols (with classical messages) with quantum adversaries.
5. With a simple construction in Section 6, we extend our basic module, in a "blind-measurement" device, where the server performs a single qubit measurement in either the Z or X basis, but he is ignorant of the measurement

basis (while the client knows). This type of blind measurement was the basis for the paper of [Mah18b], where a classical-client verification of quantum computation protocol was first given. Here we see how our module can also offer this type of functionality.

6. The Malicious 8-states QFactory can be further extended in order to offer a notion of verification for QFactory in Section 8, the new protocol being called Verifiable QFactory. We demonstrate that this notion of verifiability of QFactory is suitable, by showing that it is sufficient to obtain *verifiable blind quantum computation*. Such protocol would be the first classical client, verifiable and *blind* quantum computation protocol.
   We introduce in Subsection 8.2 a novel framework called *blind self-testing*, which differs from the standard self-testing by replacing the non-locality assumptions for such tests with blindness conditions. We describe how this technique can be used to prove the verifiability of QFactory. Note however, that the security of the Verifiable QFactory Protocol 8.2 is conjectured, while we expect that the full proof would follow using the most general case of the novel notion of *blind self-testing* that we introduced. Finally, we prove how a (much simpler) i.i.d. blind self-testing is achievable.

1.2 Overview of the protocols and proofs

The Protocol. The general idea is that a classical client communicates with a quantum server instructing him to perform certain actions. By the end of the interaction, the client obtains a random value $B = B_1 B_2 \in \{00, 01, 10, 11\}$, while the server (if he followed the protocol) ends up with the state $H^{B_1} X^{B_2} |0\rangle$, i.e. with one of the BB84 states. Moreover, the server, irrespective of whether he followed the protocol or how he deviated, cannot guess the value of the (basis) bit B_1 any better than making a random guess (more details in Subsection 3.3).

This module is sufficient to perform (either directly or with simple extensions) multiple secure computation protocols including blind quantum computation.

To achieve such a task, we require three central elements. Firstly, the quantum operations performed by the server should not be repeatable, in order to avoid letting the (adversarial) server run multiple times these operations and obtain multiple copies of the same output state. That would (obviously) compromise the security since direct tomography of a single qubit is straightforward. This can be achieved if the protocol includes a measurement of many qubits, where the probability of getting twice the same outcome would be exponentially small. The second element is that the server should not be able to efficiently classically simulate the quantum computation that he needs to perform. This is to stop the server from running everything classically and obtaining the explicit classical description of the output state. This is achieved using techniques from post-quantum cryptography and specifically the Learning-With-Errors problem. Lastly, the computation has to be easy to perform for the client, since she needs to know the output state. This asymmetry (easy for client/ hard for server) can be achieved only in the computational setting, where the client has some

extra trapdoor information. The protocol requires the following cryptographic primities defined formally in Definition 8:

- $\mathcal{F}$: a family of 2-regular, collision resistant, trapdoor one-way functions (that can be constructed from a family of injective, homomorphic, trapdoor one-way functions $\mathcal{G}$);
- $d_0(t_k)$: a hardcore predicate of the index of the functions in $\mathcal{F}$. More precisely, every function $f_k \in \mathcal{F}$ has an associated hardcore bit d_0 that is hard to guess given only k, but easy to compute given the trapdoor t_k;
- h: a predicate such that $h(x) \oplus h(x') = d_0$ for any x, x' with $f_k(x) = f_k(x')$

Given these functions, the protocol steps are: The client sends the descriptions of the functions f_k (from the family $\mathcal{F}$) and h. The server's actions are described by the circuit given in Figure 1 (see Section 3), classically instructed by the client: prepares one register at $\otimes^n H |0\rangle$ and second register at $|0\rangle^m$; then applies U_{f_k} using the first register as control and the second as target; measures the second register in the computational basis, obtains the outcome y. Through these steps server produces a superposition of the 2 preimages x and x' of y for the function f_k, i.e. $|x\rangle + |x'\rangle$. Next, server is instructed to apply the unitary corresponding to function h (targeting a new qubit $|0\rangle$) and to measure all but this new qubit in the Hadamard basis (the measurement outcomes will be denoted as b), which will be the output of the protocol. This last step intuitively magnifies the randomness of all the qubits to this final output qubit.

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

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

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

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

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

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

The Primitive Construction. In order to use this module in practise, it is crucial to have functions that satisfy our cryptographic requirements, and explore the choices of parameters that ensure that all these properties are jointly satisfied. Building on the function construction of [CCKW18] we gave specific choices that achieve these properties. The starting point is the injective, trapdoor one-way family of functions $\bar{\mathcal{G}}$ from [MP12], where the hardness of the function is derived from the Learning With Errors problem.

More precisely, to sample a function f_k, we first sample a matrix $K \in \mathbb{Z}_q^{m \times n}$ using the construction of [MP12] (that provides an injective and trapdoor function), a uniform vector $s_0 \in \mathbb{Z}_q^n$, an error $e_0 \in \mathbb{Z}_q^m$ according to a small Gaussian[3] and a random bit d_0, and we compute

$$y_0 = Ks_0 + e_0 + d_0 \times \left(\tfrac{q}{2}\ 0 \ldots 0\right)^T \tag{1.1}$$

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

$$f_{K,y_0}(s, e, c, d) = Ks + e + c \times y_0 + d \times \left(\tfrac{q}{2}\ 0 \ldots 0\right)^T \tag{1.2}$$

Note that c and d are bits, and the error e is chosen in a bigger space[4] than e_0 to ensure that the function f_{K,y_0} has two preimages with good probability. Moreover, if we define $h(s, e, c, d) = d$, it is easy to see that for all preimages x, x' with $f(x) = f(x')$, we have:

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

The Extended Protocol. In order to use the above protocol for applications such as blind quantum computing [BFK09], we need to be able to produce states taken from the (extended) set of eight states $\{|+_\theta\rangle, \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\}$. Importantly, we still need to ensure that the bits corresponding to the basis of each qubits produced, remain hidden. Here we prove how given two states produced by the basic protocol described previously, which we denote as $|\mathtt{in}_1\rangle$ and $|\mathtt{in}_2\rangle$, we can obtain a single state from the 8-states set, denoted $|out\rangle$, ensuring that no information about the bits of the basis of $|out\rangle$ is leaked[5].

To achieve this, we need to find an operation (see Figure 2 in Section 5.1), that in the honest case maps the indices of the inputs to those of the output using a map that satisfies certain conditions. This relation (inputs/output) should be such that learning anything about the basis of the output state implies learning non-negligible information for the basis of (one) input. This directly means, that

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

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

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

any computationally bounded adversary that can break the basis blindness of the output, can use this to construct an attack that would also break the basis blindness of at least one of the inputs, i.e. he would break the security guarantees of the basic module that was proven earlier.

Other Properties. To further demonstrate the utility of our core module, as a building block for other client-server protocols, one might wish to expand further the desired properties of the basic functionality. First we give a direct use of our gadget, to construct a "blind-measurement" device. Such device is essential for (non-blind) verification schemes based on post-hoc verification method [FHM18] and directly relates our work with that of [Mah18b]. Next, to obtain the verifiability of the module (i.e. imposing an honest behaviour on the server) we propose a generalization of the self-testing, where the non-locality condition is replaced by the blindness property and the analysis is done in the computational setting. Finally, to further improve the practicality of the black box call of the QFactory we also present the security against abort scenario that could be achieved based on a quantum version of Yao's XOR Lemma. However, these additional properties require stronger basic assumptions that we leave as an open question to be removed or proven correct separately.

2 Preliminaries

We assume basic familiarity with quantum notions, a good reference is [NC10]. For a state $|+_\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$, where $\theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}$, we use the notation:

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

Additionally, as L is a 3-bit string, we write it as $L = L_1L_2L_3$, where L_1, L_2, L_3 represent the bits of L.

As a result when we refer to the basis of the $|+_\theta\rangle$ state, it is equivalent to referring to the last 2 bits of L, thus saying that nothing is leaked about the basis of this state, is equivalent to saying nothing is leaked about the bits L_2 and L_3.

For a set of 4 quantum states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$, we denote the index of each state using 2 bits: B_1, B_2, with $B_1 = 0$ if and only if the state is $|0\rangle$ or $|1\rangle$, and $B_2 = 0$ if and only if the state is $|0\rangle$ or $|+\rangle$, i.e. $H^{B_1}X^{B_2}|0\rangle$. We will use interchangeably the Dirac notation and the basis/value notation.

In the following sections, we will consider polynomially bounded malicious adversaries, usually denoted by $\mathcal{A}$. The honest clients will be denoted with the π letter, and both honest parties and adversaries can output some values, that could eventually be used in other protocols. To denote that two parties π_A and $\mathcal{A}$ interact in a protocol, and that π_A outputs a while $\mathcal{A}$ outputs b, we write $(a, b) \leftarrow (\pi_A \| \pi_B)$ (we may forget the left hand side, or replace variables with underscores "_" if it is not relevant). We can also refer to the values of the classical messages send between the two parties using something like $\Pr[a = \texttt{accept} \mid (\pi_A \| \mathcal{A})]$, and this probability is implicitly over the internal randomness of π_A and $\mathcal{A}$. To specify a two-party protocol, it is enough to specify the two honest parties

(π_A, π_B). Moreover, if the protocol is just made of one round of communication, we can just write $y \leftarrow \mathcal{A}(x)$ with x the first message sent to $\mathcal{A}$, and y the messages sent from $\mathcal{A}$. Finally, a value with a tilde, such as $\tilde{d}$, represents a guess from an adversary.

We are considering protocols secure against quantum adversaries, so we assume that all the properties of our functions hold for a general Quantum Polynomial Time (QPT) adversary, rather than the usual Probabilistic Polynomial Time (PPT) one. We will denote $\mathcal{D}$ the domain of the functions, while $\mathcal{D}(n)$ is the subset of strings of length n. The following definitions are for PPT adversaries, however in this paper we will generally use quantum-safe versions of those definitions and thus security is guaranteed against QPT adversaries.

Definition 1 (One-way). *A function family $\{f_k : \mathcal{D} \to \mathcal{R}\}_{k \in \mathcal{K}}$ is **one-way** if:*

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

Definition 2 (Collision resistant). *A family of functions $\{f_k : \mathcal{D} \to \mathcal{R}\}_{k \in \mathcal{K}}$ is **collision resistant** if:*

- *There exists a PPT algorithm that can compute $f_k(x)$ for any index k, outcome of the PPT parameter-generation algorithm Gen and any input $x \in \mathcal{D}$;*
- *Any QPT algorithm $\mathcal{A}$ can find two inputs $x \neq x'$ such that $f_k(x) = f_k(x')$ with at most negligible probability over the choice of k:*
  $$\Pr_{\substack{k \leftarrow Gen(1^n) \\ rc \leftarrow \{0,1\}^*}} [\mathcal{A}(k) = (x, x') \textit{such that } x \neq x' \textit{ and } f_k(x) = f_k(x')] \leq \mathsf{negl}(n)$$
  where rc is the randomness of $\mathcal{A}$ (rc will be omitted from now).

Definition 3 (k-regular). *A deterministic function $f : \mathcal{D} \to \mathcal{R}$ is **k-regular** if $\forall y \in \operatorname{Im} f$, we have $|f^{-1}(y)| = k$.*

Definition 4 (Trapdoor Function). *A family of functions $\{f_k : \mathcal{D} \to \mathcal{R}\}$ is a **trapdoor function** if:*

- *There exists a PPT algorithm $\mathtt{Gen}$ which on input 1^n outputs (k, t_k), where k represents the index of the function. We also suppose that it is possible to derive the index k from the trapdoor t_k using a function $\mathtt{Pub}$, i.e. $k = \mathtt{Pub}(t_k)$*
- *$\{f_k : \mathcal{D} \to \mathcal{R}\}_{k \in \mathcal{K}}$ is a family of one-way functions;*
- *There exists a PPT algorithm $\mathtt{Inv}$, which on input t_k (which is called the trapdoor information) output by $\mathtt{Gen}(1^n)$ and $y = f_k(x)$ can invert y (by*

returning all preimages of y[6]) with non-negligible probability over the choice of (k, t_k) and uniform choice of x.

Definition 5 (Hardcore Predicate). *A function $hc\colon \mathcal{D} \to \{0,1\}$ is a **hardcore predicate** for a function f if:*

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

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

Definition 6 (LWE problem (informal)). *Given s, an n dimensional vector with elements in $\mathbb{Z}_q$, for some modulus q, the task is to distinguish between a set of polynomially many noisy random linear combinations of the elements of s and a set of polynomially many random numbers from $\mathbb{Z}_q$.*

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

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

Definition 7 (Function Unitary). *For any function $f : A \to B$ that can be described by a polynomially-sized classical circuit, we define the controlled-unitary U_f, as acting in the following way:*

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle \quad \forall x \in A \ \forall y \in B, \tag{2.1}$$

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

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

3 The Malicious 4-states QFactory Protocol

3.1 Requirements and protocol

The Malicious 4-states QFactory Protocol described Protocol 3.1 uses a family of cryptographic functions $\mathcal{F}$ and a function h having the following properties (see Section 4 to see how this family of functions can be constructed from a family of injective, trapdoor and (pseudo) homomorphic functions):

Definition 8 (2-regular homomorphic-hardcore family). *A family $\mathcal{F} = \{f_k : \mathcal{D}' \to \mathcal{R}\}_{k\in\mathcal{K}}$ is said to be a 2-regular homomorphic-hardcore family with respect to $h_k : \mathcal{D}' \to \{0,1\}$ and $d_0 : \mathcal{T} \to \{0,1\}$ ($\mathcal{T}$ is the set of trapdoors t_k) if:*

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

$$\Pr\left[\mathcal{A}(k) = d_0(t_k) \mid (k, t_k) \leftarrow \mathtt{Gen}_{\mathcal{F}}\right] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{3.1}$$

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

$$h_k(x) \oplus h_k(x') = d_0 \tag{3.2}$$

Note that in our specific costruction h does not depend on k, so we might omit the subscript k, and just use h, for simplicity.
We also extend this definition to δ-2-regular homomorphic-hardcore family, when the function is δ-2-regular, i.e. 2-regular with probability δ (see Definition 15 for a formal definition).

Protocol 3.1 Malicious 4-states QFactory Protocol: classical delegation of the BB84 states

---

Requirements:
Public: A δ-2-regular homomorphic-hardcore family $\mathcal{F}$ with respect to $\{h_k\}$ and d_0, as described above. For simplicity, we will represent the sets $\mathcal{D}'$ (respectively $\mathcal{R}$) using n (respectively m) bits strings: $\mathcal{D}' = \{0,1\}^n$, $\mathcal{R} = \{0,1\}^m$. In this protocol, we require δ to be negligibly close to 1, see Section 7 for an extensions to a constant δ.
Stage 1: Preimages superposition
– Client: runs the algorithm $(k, t_k) \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^n)$.
– Client: instructs Server to prepare one register at $\otimes^n H\,|0\rangle$ and second register initiated at $|0\rangle^m$.
– Client: sends k to Server and the Server applies U_{f_k} using the first register as control and the second as target.
– Server: measures the second register in the computational basis, obtains the outcome y. Here, in an honest run, the Server would have a state $(|x\rangle + |x'\rangle) \otimes |y\rangle$ with $f_k(x) = f_k(x') = y$ and $y \in \mathrm{Im}\, f_k$.
Stage 2: Output preparation
– Server: applies U_{h_k} on the preimage register $|x\rangle + |x'\rangle$ as control and another qubit

initiated at $|0\rangle$ as target. Then, measures all the qubits, but the target in the $\{\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)\}$ basis, obtaining the outcome $b = (b_1, ..., b_n)$. Now, the Server returns both y and b to the Client.

– Client: using the trapdoor t_k computes the preimages of y:

- if y does not have exactly two preimages x, x' (the server is cheating with overwhelming probability), defines $B_1 = d_0(t_k)$, and chooses $B_2 \in \{0, 1\}$ uniformly at random
- if y has exactly two preimages x, x', defines $B_1 = h_k(x) \oplus h_k(x') = d_0(t_k)$, and B_2 as defined in Theorem 2.

Output: If the protocol is run honestly, the state that the Server has produced is (with overwhelming probability) the BB84 state $|\mathtt{out}\rangle = H^{B_1} X^{B_2} |0\rangle$, having the basis $B_1 = h_k(x) \oplus h_k(x') = d_0$ (see Theorem 2 for the exact value of B_2). The output of the Server is $|\mathtt{out}\rangle$, and the output of the Client is (B_1, B_2).

3.2 Correctness of Malicious 4-states QFactory

In an honest run, the description of the output state of the protocol depends on measurement results $y \in \mathrm{Im}\, f_k$ and b, but also on the 2 preimages x and x' of y.

The output state of Malicious 4-states QFactory belongs to the set of states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ and its exact description is the following:

Theorem 2. *In an honest run, with overwhelming probability the output state $|\mathtt{out}\rangle$ of the Malicious 4-states QFactory Protocol (Protocol 3.1) is a BB84 state whose basis is $B_1 = h_k(x) \oplus h_k(x') = d_0$, and:*

- *if $d_0 = 0$, then the state is $|h_k(x)\rangle$ (computational basis, also equal to $|h_k(x')\rangle$)*
- *if $d_0 = 1$, then if $\sum_i b_i \cdot (x_i \oplus x'_i) = 0 \bmod 2$, the state is $|+\rangle$, otherwise the state is $|-\rangle$ (Hadamard basis).*

i.e.

$$|\mathtt{out}\rangle = H^{B_1} X^{B_2} |0\rangle \tag{3.3}$$

with

$$B_1 = h_k(x) \oplus h_k(x') = d_0 \tag{3.4}$$

$$B_2 = (d_0 \times (b \cdot (x \oplus x'))) \oplus h(x)h(x') \tag{3.5}$$

(the inner product is taken modulo 2, and $x \oplus x'$ is a bitwise xor)

Proof. The operations performed by the quantum server, can be described as:

$$|0\rangle \otimes |0^n\rangle \otimes |0^m\rangle \xrightarrow{I_2 \otimes H^{\otimes n} \otimes {I_2}^{\otimes m}} |0\rangle \otimes \sum_{x \in \mathcal{D}} |x\rangle \otimes |0^m\rangle \xrightarrow{I_2 \otimes U_{f_k}}$$

$$|0\rangle \otimes \sum_{x \in \mathcal{D}} |x\rangle \otimes |f_k(x)\rangle \xrightarrow{f_k \text{2-regular}} |0\rangle \otimes \sum_{y \in \mathrm{Im}(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \xrightarrow{I_2 \otimes {I_2}^{\otimes n} \otimes M_Z^{\otimes m}}$$

$$|0\rangle \otimes (|x\rangle + |x'\rangle) \otimes |y\rangle \xrightarrow{\widetilde{U}_h \otimes {I_2}^{\otimes m}} (|h(x)\rangle \otimes |x\rangle + |h(x')\rangle \otimes |x'\rangle) \otimes |y\rangle \xrightarrow{I_2 \otimes M_X^{\otimes n} \otimes {I_2}^{\otimes m}}$$

$$|\mathtt{out}\rangle \otimes |b_1\rangle ... \otimes |b_n\rangle \otimes |y\rangle \Rightarrow |\mathtt{out}\rangle = H^{d_0} X^{d_0(b \cdot (x \oplus x')) \oplus h(x)h(x')} |0\rangle$$

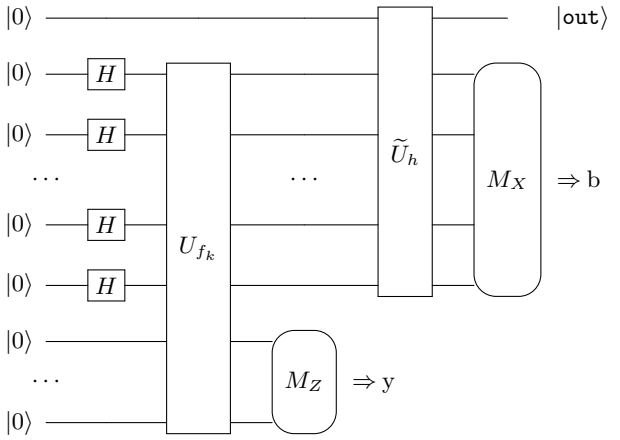

Fig. 1. The circuit computed by the Server

where $\widetilde{U}_h$ is a "swapped" U_h, acting on the first register as target and input register as control: $|0\rangle|x\rangle \xrightarrow{\widetilde{U}_h} |h(x)\rangle|x\rangle$.

The server initially prepares the state $|0^n\rangle \otimes |0^m\rangle$, where we will call the first register the preimage register, and the second one the image register.
After applying U_{f_k} we obtain the state $\sum_{x\in\mathcal{D}} |x\rangle|f_k(x)\rangle$. Using the 2-regularity property of f_k, after measuring the second register (in the computational basis) and obtaining the measurement result $y \in \mathrm{Im}(f_k)$, the state can be expressed as $(|x\rangle + |x'\rangle) \otimes |y\rangle$, where x and x' are the 2 unique preimages of y. By omitting the image register and by initializing another qubit in the $|0\rangle$ state and using the above notation, the input to the unitary $\widetilde{U}_h$ can be written as:

$$(|x\rangle + |x'\rangle) \otimes |0\rangle \tag{3.6}$$

$\widetilde{U}_h$ is basically U_h acting on the input and the new register, and after we apply it, we obtain the state:

$$(|x\rangle \otimes |h(x)\rangle + |x'\rangle \otimes |h(x')\rangle \tag{3.7}$$

As a final step, we measure all but the last qubit of this state in the $\{\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)\}$ basis (obtaining the measurement result string b), which is equivalent to applying $H^{\otimes n}$ on the input register, and then measuring it in the computational basis. Thus, after applying the Hadamard gates (which is a Fourier transformation in $\mathbb{Z}_2$), we get:

$$\sum_{b=1}^{2^n-1} (-1)^{b\cdot x} |b\rangle \otimes |h(x)\rangle + \sum_{b=1}^{2^n-1} (-1)^{b\cdot x'} |b\rangle \otimes |h(x')\rangle$$

After obtaining the measurement result b, the remaining state becomes (up to a global phase):

$$\begin{aligned}|\texttt{out}\rangle &= (-1)^{b\cdot x}\,|h(x)\rangle + (-1)^{b\cdot x'}\,|h(x')\rangle \\ &= |h(x)\rangle + (-1)^{b\cdot(x\oplus x')}\,|h(x')\rangle\end{aligned}$$

Therefore, we have:

- if $h(x) = h(x')$ (i.e. $d_0 = h(x) \oplus h(x') = 0$, using Equation 3.2), we have $|\texttt{out}\rangle = |h(x)\rangle = H^{d_0} X^{h(x)}\,|0\rangle$
- if $h(x) \neq h(x')$ (i.e. $d_0 = h(x) \oplus h(x') = 1$) we have $|\texttt{out}\rangle = |+\rangle$ iff $b\cdot(x\oplus x') = 0 \bmod 2$, and $|-\rangle$ otherwise. Thus, $|\texttt{out}\rangle = H^{d_0} X^{b\cdot(x\oplus x')}\,|0\rangle$

Hence, $|\texttt{out}\rangle = H^{B_1} X^{B_2}$ with $B_1 = d_0 = h(x) \oplus h(x')$, and

$$\begin{aligned}B2 &= (1 \oplus h(x) \oplus h(x'))h(x) + (h(x) \oplus h(x'))(b \cdot (x \oplus x')) \bmod 2 \\ &= h(x) + h(x)^2 + h(x)h(x') + d_0(b \cdot (x \oplus x')) \bmod 2 \\ &= h(x) + h(x) + h(x)h(x') + d_0(b \cdot (x \oplus x')) \bmod 2 \\ &= h(x)h(x') \oplus d_0(b \cdot (x \oplus x')) \bmod 2\end{aligned}$$

□

It can be noticed that, in an honest run of the protocol, using y and the trapdoor information of the function f_k, the Client obtains x and x' and thus can efficiently determine what is the output state that the Server has prepared. In the next section, we prove that no malicious adversary can distinguish between the 2 possible bases $\{|0\rangle, |1\rangle\}$ and $\{|+\rangle, |-\rangle\}$ of the output qubit, or equivalently distinguish whether B_1 is 0 or 1.

3.3 Security against Malicious Adversaries of Malicious 4-states QFactory

In any run of the protocol, honest or malicious, the state that the client believes that the server has is given by Theorem 2. Therefore, the task that a malicious server wants to achieve, is to be able to guess, as good as he can, the description of the output state that the client (based on the public communication) thinks the server has produced. In particular, in our case, the server needs to guess the bit B_1 (corresponding to the basis) of the (honest) output state.

Note that we want to make sure that the server cannot guess the basis bit B_1 (for most applications ([BFK09, FK17]) basis blindness is sufficient as indicated in [DK16]), and we do not care about the value bit B_2 simply because it is not possible to say that B_2 cannot be guessed with probability better than random. Indeed, even in the honest case, or in the "perfect" case with a quantum channel, the server can always measure the qubit $|\texttt{out}\rangle$ he has to extract the value bit (for example by measuring it in a random basis (computational or Hadamard) and outputting the outcome of the measurement, he will succeed with probability

$\frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times 1 = \frac{3}{4} > 1/2$). Additionally, partial blindness of B_2 is implicit in our work, since learning B_2 leads to leaking partial information about B_1, in the case that the server possesses the honest output state $H^{B_1} X^{B_2} |0\rangle$. Optimal bounds for B_2s leakage are not known if the server is malicious and without verification, is non-trivial and will be studied as a future work.

Definition 9 (4 states basis blindness). *We say that a protocol (π_A, π_B) achieves **basis-blindness** with respect to an ideal list of 4 states $S = \{S_{B_1,B_2}\}_{(B_1,B_2)\in\{0,1\}^2}$ if:*

- *S is the set of states that the protocol outputs, i.e.:*

$$\Pr\left[|\phi\rangle = S_{B_1 B_2} \in S \mid ((B_1, B_2), |\phi\rangle) \leftarrow (\pi_A \| \pi_B)\right] \geq 1 - \mathsf{negl}(n)$$

- *and no information is leaked about the index bit B_1 of the output state of the protocol, i.e for all QPT adversary $\mathcal{A}$:*

$$\Pr\left[B_1 = \tilde{B_1} \mid ((B_1, B_2), \tilde{B_1}) \leftarrow (\pi_A \| \mathcal{A})\right] \leq 1/2 + \mathsf{negl}(n)$$

Theorem 3 (Malicious 4-states QFactory is secure). *Protocol 3.1 satisfies 4-states basis blindness with respect to the ideal list of states $S = \{H^{B_1} X^{B_2} |0\rangle\}_{B_1,B_2} = \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$.*

Proof. The advantage of our construction is that this theorem is now a direct application of the definition of the family $\mathcal{F}$ (Definition 8). Indeed, let us suppose that there exists a QPT adversary $\mathcal{A}$ such that:

$$\Pr\left[B_1 = \tilde{B_1} \mid ((B_1, B_2), \tilde{B_1}) \leftarrow (\pi_A \| \mathcal{A})\right] \geq 1/2 + \frac{1}{\mathsf{poly}(n)}$$

where π_A (respectively π_B) is the honest Client (respectively Server) of Protocol 3.1. From Theorem 2, we notice that the value of B_1 is always equal to $d_0(t_k)$. Moreover, our adversary is just a one-round adversary, so we can rewrite the previous equation as:

$$\Pr\left[d_0(t_k) = \mathcal{A}(k) \mid (k, t_k) \leftarrow \mathtt{Gen}_{\mathcal{F}}\right] \geq 1/2 + \frac{1}{\mathsf{poly}(n)}$$

But d_0 is a hardcore predicate, so this contradicts Equation 3.1. So no QPT adversary $\mathcal{A}$ can guess the basis B_1 with probability better than $1/2 + \mathsf{negl}(n)$. □

Remark 1. In the run of the Malicious 4-states QFactory protocol, the adversary/server has no access to the abort/accept bit, specifying whether the Client wants to abort the protocol after receiving the image y from the server (the abort occurs when y does not have exactly two preimages). So that's why this first protocol is correct with overwhelming probability only when $\delta > 1 - \mathsf{negl}(n)$. See Section 7 to see how we address this issue for constant δ.

4 Function Implementation

4.1 General construction of 2-regular homomorphic-hardcore family

To complete the construction of Malicious 4-states QFactory, we must find functions $\mathcal{F}$, h, and d_0 satisfying the properties described in Definition 8. We first explain a general method to construct a 2-regular function from an injective homomorphic function (the generalisation to δ-2-regularity from pseudo-homomorphic functions is treated in Section I), and we give in the next section a candidate that achieves the two other properties required in our definition (homomorphic-hardcore predicate) whose security is based on the cryptographic problem LWE.

Lemma 1. *It is possible to construct a family of functions* $\mathcal{F} : \{f_{k'} : \mathcal{D} \times \{0,1\} \to \mathcal{R}\}$, $h'_{k'}$ *and* d_0 *that are a 2-regular homomorphic-hardcore family (Definition 8) from a family of functions* $\mathcal{G} = \{g_k : \mathcal{D} \to \mathcal{R}\}_k$ *that is:*

- *injective*
- *trapdoor*
- *homomorphic* [7]

and such that there exists a homomorphic hardcore predicate $h_k : \mathcal{D} \to \{0,1\}$ *for all* $g_k \in \mathcal{G}$.

Proof. Because $\mathcal{G}$ is homomorphic, there exist 2 operations " $+_{\mathcal{D}}$ " acting on $\mathcal{D}$ and " $+_{\mathcal{R}}$ " acting on $\mathcal{R}$ such that:

$$g_k(z_1 +_{\mathcal{D}} z_2) = g_k(z_1) +_{\mathcal{R}} g_k(z_2) \ \forall k \ \forall z_1, z_2 \in \mathcal{D} \tag{4.1}$$

The function $h : \mathcal{D} \to \{0,1\}$ is homomorphic, so:

$$h(z_1) \oplus h(z_2) = h(z_2 +_{\mathcal{D}} z_1) \ \forall z_1, z_2 \in \mathcal{D} \tag{4.2}$$

and because we are working modulo 2 it is easy to see that:

$$h(z_1) \oplus h(z_2) = h(z_2 -_{\mathcal{D}} z_1) \ \forall z_1, z_2 \in \mathcal{D}, \tag{4.3}$$

where "$-_{\mathcal{D}}$" is the inverse of the operation "$+_{\mathcal{D}}$".

Then, the functions $\mathcal{F}$, h'_k, d_0 are constructed as follow. First, to generate a private key, we generate a private key of $\mathcal{G}$, and we pick a random element z_0:

[7] We only require $\mathcal{G}$ to be homomorphic with good probability for a single application of the operation $+_{\mathcal{D}}$ and this would result in $\mathcal{F}$ being 2-regular with good probability, as proven in Section I.

$\underline{\mathtt{Gen}_{\mathcal{F}}(1^n)}$

$(k, t_k) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$

$z_0 \leftarrow_{\$} \mathcal{D}$

$y_0 = g_k(z_0)$

$t'_{k'} = (t_k, z_0)$

$k' = (k, y_0)$

return $(k', t'_{k'})$

And then we define $f_{k'} : \mathcal{D} \times \{0,1\} \to \mathcal{R}$ as:

$$f_{k'}(z, c) = g_k(z) +_{\mathcal{R}} c \cdot y_0$$

We also define $h'_{k'} : \mathcal{D} \times \{0,1\} \to \mathcal{R}$ as:

$$h'_{k'}(z, c) = h_k(z)$$

and $d_0 : \mathcal{T}' \to \{0,1\}$ (where $\mathcal{T}'$ is the sets of trapdoors $t'_{k'}$) as

$$d_0(t_k, z_0) = h_k(z_0)$$

Now we need to check the properties of $\mathcal{F}$, $h'_{k'}$ and d_0:

- the 2-regularity of $\mathcal{F}$ comes directly from the injectivity of $\mathcal{G}$: for any $y \in \operatorname{Im} f_k$, we have one preimage $(g^{-1}(y), 0)$ and one preimage $(g^{-1}(y) - z_0, 1)$. The past two formula also show the trapdoor property using the fact that $\mathcal{G}$ is a trapdoor family (more details can be found in Theorem 6.1, in [CCKW18]).
- the collision-resistant property comes from the homomorphicity , injectivity, and one-wayness of $\mathcal{G}$: if we find a collision, we can write a reduction that breaks the one-wayness of $\mathcal{G}$. Indeed by injectivity two different preimages have a different c, i.e. $f_k(z, 0) = f_k(z', 1)$. So $g_k(z) = g_k(z') + g_k(z_0)$, i.e. $z_0 = z - z'$. So it means that from a collision we can find z_0, which is absurd because g_k is one way.
- $h'_{k'}$ is equal to h_k, so it is possible to compute it efficiently
- $d_0(t'_{k'}) = h_k(z_0)$, and h_k is a hardcore predicate, so d_0 is also hardcore
- The last condition is respected, because if $f_k(z, 0) = f_k(z', 1)$,

$$\begin{aligned} h'_{k'}(z, 0) \oplus h'_{k'}(z', 1) &= h_k(z) \oplus h_k(z') \\ &= h_k(z -_{\mathcal{R}} z') \\ &= h_k(z_0) \\ &= d_0(t'_{k'}) = d_0 \end{aligned}$$

□

4.2 Construction of δ-2-regular homomorphic-hardcore family $\mathcal{F}$

We will now give an explicit implementation of a family $\mathcal{G}$ that is injective, trapdoor, (pseudo) homomorphic with a homomorphic-hardcore predicate d_0,

and then we will rely on a construction similar to Lemma 1 to produce a family $\mathcal{F}$, h, and d_0 with the properties described in Definition 8 needed by Protocol 3.1 and Protocol 7.1. Note that we defined in a previous work [CCKW18] a similar construction, but without the additional homomorphic-hardcore property.

The starting point is the injective, trapdoor one-way family of functions $\bar{\mathcal{G}} = \{\bar{g}_K : \mathbb{Z}_q^n \times E^m \to \mathbb{Z}_q^m\}_K$[8] from [MP12] (where E defines the set of integers bounded in absolute value by some "big-enough" value μ which will be defined later, and additions are matrix additions modulo q, where q is an even integer).

$$\bar{g}_K(s, e) = Ks + e$$

Then, to sample a function from the family $\mathcal{F} = \{f_k : \mathbb{Z}_q^n \times E^m \times \{0,1\} \times \{0,1\} \to \mathbb{Z}_q^m\}$, we will first sample a random matrix $K \in \mathbb{Z}_q^{m \times n}$ with the trapdoor matrix R using the construction from [MP12], as well as a uniform random vector $s_0 \in \mathbb{Z}_q^n$, a random small error vector $e_0 \in \mathbb{Z}_q^m$ sampled according to a "small-enough" Gaussian distribution $\mathcal{D}_{\alpha' q}^m$ on integers and a (uniform) random bit $d_0 \in \{0,1\}$. Now, after defining the constant vector $v = \left(\frac{q}{2}\ 0 \ldots 0\right)^T$, and

$$y_0 := Ks_0 + e_0 + d_0 \times \begin{pmatrix} \frac{q}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = Ks_0 + e_0 + d_0 \times v$$

the trapdoor is set to $t_k := (R, s_0, e_0, d_0)$, and the public index is $k = (K, y_0)$. We can already note at that step that d_0 is a hardcore-predicate:

Lemma 2. *The function $d_0(t_k) := d_0$ is a hardcore predicate of k, i.e. for all QPT adversaries $\mathcal{A}$,*

$$\Pr[\mathcal{A}(k) = d_0(t_k)] \leq \frac{1}{2} + \mathsf{negl}(n)$$

The proof of Lemma 2 is in Section A.
Now, we can define $f_k : \mathbb{Z}_q^n \times E^m \times \{0,1\} \times \{0,1\} \to \mathbb{Z}_q^m$ as follow:

$$f_k(s, e, c, d) = Ks + e + c \times y_0 + d \times v$$

and $h : \mathbb{Z}_q^n \times E^m \times \{0,1\} \times \{0,1 \to \{0,1\}$ as:

$$h(s, e, c, d) = d$$

The intuition behind this construction is more or less the same as the general construction presented in Subsection 4.1. Moreover, the first two terms $As + e$ are useful for the security, the $c \times y_0$ term is needed to ensure the 2-regularity (the two images will differ by $(s_0, e_0, 1, d_0)$), and the last term $d \times v$ is mostly useful to provide the hardcore property. More precisely:

[8] The bar on top of $\bar{\mathcal{G}}$ denotes the version where there is not yet the hardcore bit d_0

– This function cannot have more than 2 preimages because the partial functions $f(\cdot,\cdot,c,\cdot)$ are injective (because $\bar{g}_K$ is injective)
– h is the homomorphic-hardcore predicate required by Definition 8. Indeed, if there is a collision, i.e. if $f_k(s,e,0,d) = f_k(s',e',1,d')$, it is easy to see that $d \oplus d' = d_0$ (q is even, and operations are modulo q), i.e. that $h(x) \oplus h(x') = d_0$
– finally, for an appropriate choice of parameters (see Lemma 3), this function is 2-regular with good probability. Indeed, if for a random element $(s,e,0,d)$ there exists $(s',e',1,d')$ with $f_k(s,e,0,d) = f_k(s',e',1,d')$, then $e = e' + e_0$. But e_0 is sampled from a set significantly smaller than E, so with good probability $e' = e - e_0$ will belong to E.

Note on the parameters: α' is chosen to make sure that the sampled elements are small compared to μ (the upper bound on E), but such that the noise is still big enough for security. On the contrary, μ must stay small enough to ensure that the function does not have more than two preimages. Our previous work provides a set of parameters having all the required constraints:

Lemma 3 (from [CCKW18]). *The family of functions $\mathcal{F}$ is δ-2-regular with good (constant greater than 1/2) probability, trapdoor, one-way and collision resistant (all these properties are true even against a quantum attacker), assuming that there is no quantum algorithm that can efficiently solve* SIVP_γ *for* $\gamma = \mathsf{poly}(n)$*, for the following choices of parameters:*

$$\begin{aligned} q &= 2^{5\lceil log(n) \rceil + 21} \\ m &= 23n + 5n\lceil log(n) \rceil \\ \mu &= 2mn\sqrt{23 + 5log(n)} \\ \alpha' &= \frac{\mu}{m\sqrt{m}q} \end{aligned} \tag{4.4}$$

Moreover, we can find another set of parameters such that this probability δ is negligibly close to one assuming that SIVP_γ *is secure for a superpolynomial γ (depending on the value of δ, you may choose Protocol 3.1 ($\delta \sim 1$) or Protocol 7.1 ($\delta > 1/2$)).*

We can now formalize the above intuitions:

Theorem 4. *The family $\mathcal{F}$ defined above with appropriate parameters such as the one defined in Lemma 3 is a δ-2-regular homomorphic-hardcore family.*

Proof. The proofs that $h_k(x) \oplus h_k(x') = d_0$, and that g_K is injective and one-way can be found in Section A, the Lemma 3 ensures that the family is δ-2-regular, the hardcore property comes from Lemma 2, and the other properties are trivial to check. □

5 The Malicious 8-states QFactory Protocol

In order to use the Malicious 4-states QFactory Protocol functionality for applications such as blind quantum computing [BFK09], we need to be able to

produce states taken from the set $\{|+_\theta\rangle, \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\}$, always ensuring that the bases of these qubits remain hidden. Here we prove how by obtaining two states of Malicious 4-states QFactory Protocol, we can obtain a single state from the 8-states set, while no information about the bases of the new output state is leaked.

To achieve this, we need to find an operation, that in the honest case maps the correct inputs to the outputs, in such a way, that the index of the output state corresponding to the basis, is directly related with the bases bits of the input states. This relation should be such that learning anything about the basis of the output state implies learning non-negligible information about the input. This directly means, that any computationally bounded adversary that breaks the 8-states basis blindness of the output, also breaks the 4-states basis blindness of at least one of the inputs.[9]

Protocol 5.1 Malicious 8-states QFactory

Requirements: Same as in Protocol Protocol 3.1

Input: Client runs 2 times the algorithm $Gen_{\mathcal{F}}(1^n)$, obtaining $(k^1, t_k^1), (k^2, t_k^2)$. Client keeps t_k^1, t_k^2 private.

Protocol:

– Client: runs Malicious 4-states QFactory algorithm to obtain a state $|\mathtt{in}_1\rangle$ and a "rotated" Malicious 4-states QFactory to obtain a state $|\mathtt{in}_2\rangle$ (by a rotated Malicious 4-states QFactory we mean a Malicious 4-states QFactory, but where the last set of measurements in the $|\pm\rangle$ basis (Figure 1) is replaced by a set of measurements in the $\left|\pm\frac{\pi}{2}\right\rangle$ basis).

– Client: records measurement outcomes (y^1, b^1), (y^2, b^2) and computes and stores the corresponding indices of the output states of the 2 Malicious 4-states QFactory runs: (B_1, B_2) for $|\mathtt{in}_1\rangle$ and (B_1', B_2') for $|\mathtt{in}_2\rangle$.

– Client: instructs Server to apply the Merge Gadget (Figure 2) on the states $|\mathtt{in}_1\rangle$, $|\mathtt{in}_2\rangle$.

– Server: returns the 2 measurement results s_1, s_2.

– Client: using (B_1, B_2), (B_1', B_2'), s_1, s_2 computes the index $L = L_1L_2L_3 \in \{0,1\}^3$ of the output state.

Output: If the protocol is run honestly, the state that the Server has produced is:

$$|\mathtt{out}\rangle = X^{(s_2+B_2)\cdot B_1} Z^{B_2'+B_2(1-B_1)+B_1[s_1+(s_2+B_2)B_1']} R(\frac{\pi}{2})^{B_1} R(\frac{\pi}{4})^{B_1'} |+\rangle \qquad (5.1)$$

5.1 Correctness of Malicious 8-states QFactory

We prove the existence of a mapping $\mathcal{M}$ (which we will call Merge Gadget), from 2 states $|\mathtt{in}_1\rangle$ and $|\mathtt{in}_2\rangle$, where $|\mathtt{in}_1\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ and $|\mathtt{in}_2\rangle \in$

[9] Here it is worth pointing out that a similar result (in a more complicated method) was achieved in [DK16]. That technique however, is applied in the information theoretic setting.

$\{|+\rangle, |-\rangle, |+_y\rangle, |-_y\rangle\}$ to a state $|\mathtt{out}\rangle = \left|+_{L\cdot\frac{\pi}{4}}\right\rangle$, where $L = L_1L_2L_3 \in \{0,1\}^3$. Namely, as defined in Protocol 5.1, $\mathcal{M}$ is acting in the following way:

$$\mathcal{M}(|\mathtt{in}_1\rangle, |\mathtt{in}_2\rangle) = M_{X,2}M_{X,1}\wedge Z_{2,3}\wedge Z_{1,2}\left[\left|+_{\frac{\pi}{4}}\right\rangle \otimes |\mathtt{in}_1\rangle \otimes |\mathtt{in}_2\rangle\right] \quad (5.2)$$

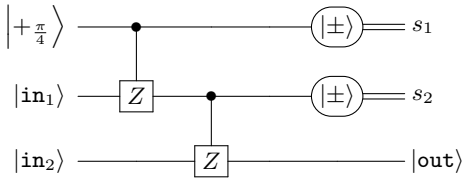

Fig. 2. Merge Gadget

Theorem 5. *In an honest run, the Output state of the Malicious 8-states QFactory Protocol is of the form* $\left|+_{L\cdot\frac{\pi}{4}}\right\rangle$*, where* $L = L_1L_2L_3 \in \{0,1\}^3$.

Proof. In an honest run, using the Merge Gadget (Figure 2) we get:

$$\mathcal{M}(|\mathtt{in}_1\rangle, |\mathtt{in}_2\rangle) = M_{X,2}M_{X,1}\wedge Z_{2,3}\wedge Z_{1,2}\left[\left|+_{\frac{\pi}{4}}\right\rangle \otimes |\mathtt{in}_1\rangle \otimes |\mathtt{in}_2\rangle\right] \quad (5.3)$$

Using the correctness of Malicious 4-states QFactory (Thereom 2), we have that:

$$|\mathtt{in}_1\rangle = H^{B_1}X^{B_2}|0\rangle \quad (5.4)$$

$$|\mathtt{in}_2\rangle = R(\frac{\pi}{2})^{B_1'}Z^{B_2'}|+\rangle \quad (5.5)$$

Thus:

$$|\mathtt{out}\rangle = M_{X,2}M_{X,1}\wedge Z_{2,3}\wedge Z_{1,2}\left[\left|+_{\frac{\pi}{4}}\right\rangle \otimes H^{B_1}Z^{B_2}|+\rangle \otimes R(\frac{\pi}{2})^{B_1'}Z^{B_2'}|+\rangle\right] \quad (5.6)$$

Which is then equivalent to:

$$|\mathtt{out}\rangle = R[\pi(B_2' + B_2 + B_1\cdot(s_1+s_2)) + \frac{\pi}{2}(B_1' + (B_2+s_2)\cdot B_1) + \frac{\pi}{4}B_1]|+\rangle \quad (5.7)$$

As a result, we obtain:

$$L_1 = B_2' \oplus B_2 \oplus [B_1\cdot(s_1\oplus s_2)]$$

$$L_2 = B_1' \oplus [(B_2\oplus s_2)\cdot B_1] \quad (5.8)$$

$$L_3 = B_1 \quad (5.9)$$

□

It can also be noticed that, in an honest run of Malicious 8-states QFactory, the client can efficiently determine L: using b^1, b^2, y^1, y^2 and the trapdoors t_k^1, t_k^2, he first obtains (B_1, B_2) and (B_1', B_2'), and after receiving s_1, s_2, he determines the description of the state prepared by the server.

5.2 Security against Malicious Adversaries of Malicious 8-states QFactory

In any run of the protocol, honest or malicious, the state that the client believes that the server has, is given by Theorem 5.
Therefore, as in the case of Malicious 4-states QFactory, the task that a malicious server wants to achieve, is to be able to guess, as good as he can, the index of the output state that the client thinks the server has produced. In particular, in our case, the server needs to guess the bits L_2 and L_3 (corresponding to the basis) of the (honest) output state.

Definition 10 (8 states basis blindness). *Similarly, we say that a protocol* (π_A, π_B) *achieves* ***basis-blindness*** *with respect to an ideal list of 8 states* $S = \{S_{L_1,L_2,L_3}\}_{(L_1,L_2,L_3)\in\{0,1\}^3}$ *if:*

- *S is the set of states that the protocol outputs, i.e.:*

$$\Pr\left[|\phi\rangle = S_{L_1,L_2,L_3} \in S \mid ((L_1,L_2,L_3),|\phi\rangle) \leftarrow (\pi_A \| \pi_B)\right] = 1$$

- *and if no information is leaked about the "basis" bits (L_2, L_3) of the output state of the protocol, i.e for all QPT adversary $\mathcal{A}$:*

$$\Pr\left[L_2 = \tilde{L_2} \text{ and } L_3 = \tilde{L_3} \mid ((L_1,L_2,L_3),(\tilde{L_2},\tilde{L_3})) \leftarrow (\pi_A \| \mathcal{A})\right] \leq 1/4 + \mathsf{negl}(n)$$

Theorem 6. *Malicious 8-states QFactory satisfies 8-state basis blindness with respect to the ideal set of states* $S = \{|+_{\pi L/4}\rangle\}_{L\in\{0,\ldots,7\}} = \{|+\rangle, |+_{\frac{\pi}{4}}\rangle, .., \left|+_{\frac{7\pi}{4}}\right\rangle\}$.

Sketch Proof (The full proof can be found in Section B.). We prove this result by reduction showing that, if there exists a QPT adversary $\mathcal{A}$ that is able to break the 8-states basis blindness property of Malicious 8-states QFactory (determine the indices L_2 and L_3 with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$ for some polynomial function $poly_1$), then we can construct a QPT adversary $\mathcal{A}'$ that can break the 4-states basis blindness of the Malicious 4-states QFactory protocol (determine the basis bit with probability $\frac{1}{2} + \frac{1}{poly_2(n)}$, for some polynomial $poly_2(\cdot)$). □

6 Blind Measurement Gadget

In this section we show how our basic module can be used to achieve another task, that of blind-measurement. The task is the following: we want a classical client to instruct a quantum server to measure one of his qubits in either the Pauli X or Z basis, in a way that the server is not aware of which of the two bases was actually used. We give below a gadget that achieves this task using a single output of the Malicious QFactory 4-states. We note also, that other blind-measurements, between different sets of bases, are also possible using our module, but we focus on this one since it is this type of measurement needed for post-hoc verification [FHM18] and thus is the basis for classical verification protocols such as that of [Mah18b].

The following gadget Fig. 3 achieves the desired task. Note, that we consider a general state $|\psi\rangle$ where the measurement is performed on the first of its qubits (thus the second wire in the figure represents all the non-measured qubits). Depending on the value of B_1, the actual outcome is Z or X and the measurement outcome is obtained from either s_1 or s_2. See details in Section D.

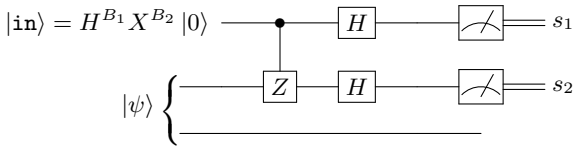

Fig. 3. Blind Measurement Gadget

7 Malicious-abort 4-states QFactory: treating abort case

In this section, we will discuss an extension of Malicious 4-states QFactory, whose aim is to achieve basis blindness even against adversaries that try to exploit the fact that Malicious 4-states QFactory can abort when there is only one preimage associated to the y returned by the server. One may think that we could just send back this `accept`/`abort` bit to the server, but unfortunately it could leak additional information on the hardcore bit d_0 (which corresponds to the basis B_1 of the produced qubit) to the server, and from an information theory point of view, as soon as the probability of acceptance is small enough, we cannot guarantee that this bit remains secret. On the other hand, for honest servers, the probability of aborting is usually non-negligible, so we cannot neglect this case.

We stress out that it is also possible to guarantee that for honest servers this probability goes negligibly close to 1 by making an appropriate choice of parameters for the function. In that case the initial protocol of Malicious QFactory defined Section 3 is secure, but this comes (as far as we know), at the cost of using a function with is "less" secure. More specifically, instead of having a reduction to GapSVP with a polynomial γ, the reduction usually goes to GapSVP with a super-polynomial γ. Such function parameters have been used implicitly in other works [Mah18a] ([Bra18] later removed this assumption), and for now they are believed to be secure (the best known polynomial algorithm cannot break GapSVP with a γ smaller than exponential), but these assumptions are usually not widely accepted in the cryptography community, and that's why we aim to remove this non-standard assumption.

The solution we propose in this section uses the assumption that the classical Yao's XOR Lemma also applies for one-round protocols (with classical messages) against quantum adversary. This lemma roughly states that if you cannot guess the output bit of one round with probability better than η, then it's hard to

guess the output bit of t independents rounds with probability much better than $1/2 + \eta^t$. As far as we know, this lemma has been proven only in the classical case (see [GNW11] for a review of this theorem as well as the main proof methods), and other works [VW08] even extend this lemma to protocols, and also to quantum setting [She10, KSd04]. Unfortunately, these works focus on communication and query complexity and are not really usable in our case.

Note also that we are also working on some other proof methods to get rid of this last assumption and to improve the efficiency of the protocol by avoiding repetition (see Section K for more details).

In the following, we will call "accepted run" a run of Malicious 4-states QFactory such that the y received from the server has 2 preimages ("probability of success" also refers to the probability of this event when the server is honest), and otherwise we call it an "aborted run".

7.1 The Malicious-Abort 4-state QFactory Protocol

In a nutshell, the solution we propose is to run several instances of Malicious 4-states QFactory, by remarking that we do not need to discard the aborted runs. Indeed, it is easy to see that in these cases, the produced qubits will always be in the same basis (denoted by 0). The idea is then to implement on the server side a circuit that will output a qubit having as basis the XOR of all the basis of the accepted runs (without even leaking which runs are accepted or not), and check on client's side that the number of accepted runs is high enough (this will happen with probability exponentially close to 1 for honest servers). If it is the case, the client will just output the XOR of the basis of the accepted run, and otherwise (i.e. if the server is malicious), he will just pick a random bit value.

Unfortunately, in practice things are a bit more complicated, and in order to be able to write the proof of security we need to divide all the t runs into n_c "chunks" of size t_c, and test them individually. Here is a more precise (but still high level) description of the protocol and proof's ideas:

- firstly, we run $t = n_c \times t_c$ parallel instances of Malicious 4-states QFactory, without revealing the abort bit for any of these instances;
- then the key point to note is that for honest servers, if y_i has only one preimage then the output qubit produced by the server at the end of the protocol will be either $|0\rangle$ or $|1\rangle$, but cannot be $|+\rangle$ or $|-\rangle$ (with one preimage we do not have a superposition). In other words, the basis is always the $\{|0\rangle, |1\rangle\}$ basis (denoted as 0) so we do not really need to abort. Therefore, at the end, (for honest runs) the basis of the output qubits will be equal for all $i \in [\![0, t]\!]$ to $\beta_i = d_{0,i} \cdot a_i$, where $a_i = 1$ iff y_i has two preimages, and $a_i = 0$ otherwise. Of course, this distribution will be biased against 0, but it is not a problem. See Lemma 4 for proof.
- then, it also appears that from t qubits in the basis $\beta_1, \ldots, \beta_t$, we have a way to produce a single qubit belonging to the set $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ whose basis B_1 is the XOR of the basis of the t qubits, i.e. $B_1 = \oplus_{i=1}^{t} \beta_i$ (see Lemma 5).

– Then, the client will test every chunk, by checking if the proportion of accepted runs in every chunk is greater than a given value p_c. If all chunks have enough accepted runs, then the client just computes and outputs the good value for the basis (which is the XOR of the hardcore bit of all the accepted runs) and value bits. However, if at least one chunk doesn't have enough accepted runs (which shouldn't happen if the server is honest), then the client just outputs random values for the basis and value bit, not correlated with server's qubit (equivalent to saying that a malicious server can always throw the qubit and pick a new qubit, not correlated with client's one).
– Correctness: if the probability to have two preimages for an honest server is at least a constant p_a greater than $1/2$ (the parameters we proposed in [CCKW18] have this property), and if t is chosen high enough, the fraction of accepted runs will be close to p_a, and we can show that the probability to have a fraction of accepted runs smaller than a given constant $p_b < p_a$ is exponentially (in t) close to 0 (cf Lemma 6). So with overwhelming probability, all the chunks will have enough accepted runs, i.e. honest servers will have a qubit corresponding to the output of the client.
– Soundness: to prove the security of this scheme, we first prove Lemma 8 that it is impossible for any adversary to guess the output of one chunk with a probability bigger than a constant $\eta < 1$ (otherwise we have a direct reduction that breaks the hardcore bit property of g_K). Now, using the quantum version of Yao's XOR Lemma that we conjecture at Conjecture 1, we can deduce that no malicious server is able to guess the XOR of the t_c chunks/instances with probability better than $1/2 + \eta^{t_c} + \mathsf{negl}(n)$, which goes negligibly close to $1/2$ when $t_c = \Omega(n)$.

So putting everything together, the parties will just run $t = n_c \cdot t_c$ Malicious 4-states QFactory in parallel, the client will then check if $\sum_i a_i$ is higher than $p_c \cdot t_c$ for all the n_c chunks, and if so he will set $B_1 = \oplus_{i=1}^{t} d_i \cdot a_i$ (server has a circuit to produce a qubit in this basis as well). Otherwise B_1 will be set to a uniformly chosen random bit (it is equivalent to say that a malicious server can destroy the qubit, and this is also unavoidable even with a real quantum communication), and we still have correctness with overwhelming probability for honest clients.

The exact algorithm is described in Protocol 7.1, while the security result is shown in Theorem 7 (and the proofs can be found in Appendix J).

7.2 Correctness and security of Malicious-Abort 4-state QFactory

Now, we will formalize and prove the previous statements.

Conjecture 1 (Yao's XOR Lemma for one-round protocols (classical messages) against quantum adversary).
Let n be the security parameter, let $f_n : \mathcal{X}_n \times \mathcal{Y}_n \to \{0,1\}$ be a (possibly non-deterministic) family of functions (usually not computable in polynomial time), and let χ_n be a distribution on $\mathcal{X}_n$ efficiently samplable. If there exists $\delta(n)$ such

that $|\delta(n)| \geq \frac{1}{\mathsf{poly}(n)}$ and such that for all polynomial (in n) quantum adversary $\mathcal{A}_n : \mathcal{X}_n \to \mathcal{Y}_n \times \{0,1\}$,

$$\Pr\left[\tilde{\beta} = f_n(x,y) \mid (y,\tilde{\beta}) \leftarrow \mathcal{A}_n(x), x \leftarrow \chi_n\right] \leq 1 - \delta(n)$$

then, for all $t \in \mathbb{N}^*$, there is no QPT adversary $\mathcal{A}'_n : \mathcal{X}_n^t \to \mathcal{Y}_n^t \times \{0,1\}$ such that:

$$\begin{aligned}&\Pr\left[\tilde{\beta} = \bigoplus_{i=1}^{t} f_n(x_i,y_i) \mid (y_1,\ldots,y_t,\tilde{\beta}) \leftarrow \mathcal{A}'_n(x_1,\ldots,x_t), \forall i, x_i \leftarrow \chi_n\right]\\ \geq &\frac{1}{2} + (1-\delta(n))^t + \mathsf{negl}(n)\end{aligned}$$

Lemma 4 (Aborted runs are useful). *If π_{A_4} and π_{B_4} are following the Malicious 4-states QFactory protocol honestly, and if y has not 2 preimages, then the output qubit produced by π_{B_4} is in the basis $\{|0\rangle, |1\rangle\}$.*

See proof in Section J.

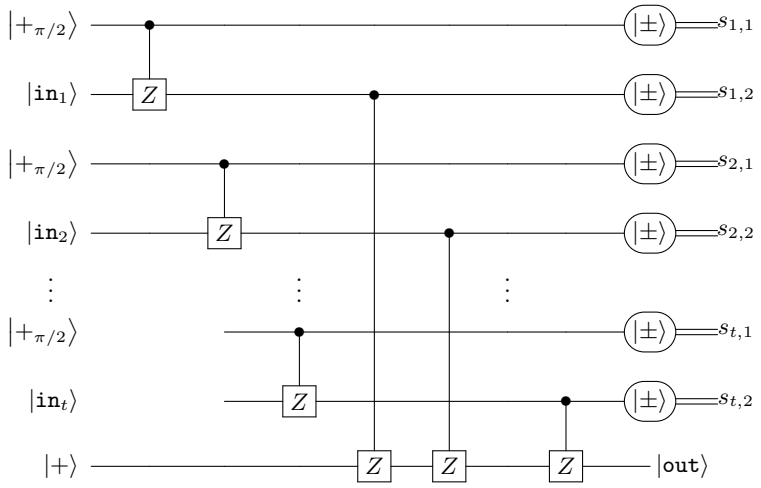

Fig. 4. The XOR gadget circuit $\mathtt{Gad}_{\oplus}$ (run on server side)

Lemma 5 (Gadget circuit $\mathtt{Gad}_{\oplus}$ computes XOR). *If we denote by b_i the basis of $|\mathtt{in}_i\rangle$ (equal to 0 if the basis is 0/1, and 1 if the basis is $+/-$), and if we run the circuit $\mathtt{Gad}_{\oplus}$ (inspired by measurement based quantum computing) represented Figure 4 on these inputs, then basis of $|\mathtt{out}\rangle$ is equal to $\oplus_{i=1}^{t} b_i$.*

See proof in Section J.

We will now describe here the protocol of Malicious-Abort 4-states QFactory:

Protocol 7.1 Malicious-Abort 4-states QFactory Protocol

Requirements:
Public: The family of functions $\mathcal{F}$ and h described above, such that the probability of having two pre-images for a random image is greater than a constant $p_a > 1/2$.
This protocol is based on the constants $t_c \in \mathbb{N}$ (number of repetitions per chunk), $n_c \in \mathbb{N}$ (number of chunks), $p_a \in (1/2, 1]$ (lower bound on probability of accepted run in the honest protocol), $p_c \in (1/2, 1] < p_a$ (fraction of the runs per chunk that must be accepted). These constants can be chosen to have overwhelming probability of success for honest players, and negligible advantage for an adversary trying to guess the basis.
Stage 1: Run multiple QFactories
– Client: prepares $t = n_c \times t_c$ public keys/trapdoors:

$$\left((k^{(i,j)}, t_{k^{(i,j)}}) \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^n)\right)_{i \in [\![1,n_c]\!], j \in [\![1,t_c]\!]}$$

The Client then sends the public keys $k^{(i,j)}$ to the Server, together with h.
– Server and Client: follow Protocol 3.1 t times, with the keys sent at the step before. Client receives $((y^{(i,j)}, b^{(i,j)}))_{i,j}$, and sets for all i, j: $a^{(i,j)} = 1$ iff $|f^{-1}(y^{(i,j)})| = 2$, otherwise $a^{(i,j)} = 0$, and $B_1^{(i,j)}$ and $B_2^{(i,j)}$ like in Protocol 3.1 when $a^{(i,j)} = 1$ (otherwise $B_1^{(i,j)} = 0$ and $B_2^{(i,j)} = h(f^{-1}(y))$. Server will get t outputs $\left|\mathtt{in}_{(i,j)}\right\rangle$.
Stage 2: Combine runs and output
– Server: applies circuit Figure 4 on the t outputs $|\mathtt{in}_t\rangle$, and outputs $|\mathtt{out}\rangle$.
– Client: checks that for all chunks $i \in [\![1, n_c]\!]$ the number of accepted runs is high enough, i.e. $\sum_j a^{(i,j)} \geq p_c t_c$.

- If at least one chunk does not respect this condition, then picks two random bits B_1 (the basis bit) and B_2 (the value bit) and outputs (B_1, B_2), corresponding to the description of the BB84 state $H^{B_1} X^{B_2} |0\rangle$.
- If all chunks respect this condition, then sets $B_1 := \bigoplus_{i,j} B_1^{(i,j)}$ (the final basis is the XOR of all the basis), and B_2 will be chosen to match the output of Figure 4.

Lemma 6 (Probability of correctness of Malicious-Abort 4-states QFactory for one chunk). *If the probability to have an accepted run in Malicious 4-states QFactory with honest parties is greater than a constant $p_a > 1/2$, i.e.*

$$\Pr\left[|f_k^{-1}(y)| = 2 \mid (\pi_{A_4} \| \pi_{B_4})\right] \geq p_a$$

(where π_{A_4} and π_{B_4} are the honest protocols of Malicious 4-states QFactory) then the probability to have at least $p_b t_c$ accepted runs (with $p_b < p_a$, p_b considered as a constant) is exponentially (in t_c) close to 1:

$$\Pr\left[\sum_i a_i \geq p_b t_c \mid (\pi_{A_4 \oplus c}^{t_c} \| \pi_{B_4 \oplus c}^{t_c})\right] \geq 1 - \frac{1}{e^{2(p_a - p_b)^2 t_c}} = 1 - \mathsf{negl}(t_c)$$

(where $\pi_{A_4 \oplus c}^{t_c}$ and $\pi_{B_4 \oplus c}^{t_c}$ are the (honest) parties of the Protocol 7.1 restricted on one chunk of size t_c, or, equivalently t_c parallel repetitions of Protocol 3.1)

See proof in Section J.

Lemma 7 (Correctness of Protocol 7.1). *The protocol Protocol 7.1 is correct with overwhelming probability as soon as $t = \mathsf{poly}(n)$ and $t_c = \Omega(n)$, i.e.*

$$\Pr\left[|\mathtt{out}\rangle = H^{B_1} Z^{B_2} \mid ((B_1, B_2), |\mathtt{out}\rangle) \leftarrow (\pi_A \| \pi_B)\right] \geq 1 - \mathsf{negl}(n)$$

See proof in Section J.

Definition 11. *For any public key k and image y, we define $a(k, y) = 1$ iff $|f_k^{-1}(y)| = 2$, and $a(k, y) = 0$ otherwise.*
Then, for all $t_c \in \mathbb{N}$ and $p_c \in [0, 1]$, we define $\beta_{t_c,p_c}(k^{(1)}, \ldots, k^{(t_c)}, y^{(1)}, \ldots, y^{(t_c)})$ as the (randomized) function that outputs a random bit if $\sum_i a(k^{(i)}, y^{(i)}) < p_c \cdot t_c$, and outputs otherwise $\oplus_i (a(k^{(i)}, y^{(i)}) \cdot d_0^{(i)})$, where $d_0^{(i)}$ is the hardcore bit corresponding to $k^{(i)} := (K^{(i)}, g_{K^{(i)}}(z_0^{(i)}))$, i.e. $d_0^{(i)} = h(z_0^{(i)})$.

Lemma 8 (Solving one chunk is difficult). *Let $p_c \in (\frac{1}{2}, 1]$. Then, there exists no polynomial adversary $\mathcal{A}$ such that:*

$$\Pr\Big[\tilde{B}_1 = \beta_{t_c,p_c}(k^{(1)}, \ldots, k^{(t_c)}, y^{(1)}, \ldots, y^{(t_c)}) \mid (y^{(1)}, \ldots, y^{(t_c)}, \tilde{B}_1) \leftarrow \mathcal{A}(k^{(1)}, \ldots, k^{(t_c)})\Big] > \eta$$

with $\eta = \frac{1}{2}\left(1 + \frac{1}{2p_c}\right)$, where the randomness is over the randomness of β, $\mathcal{A}$, and over the choice of $(k^{(i)})_i$ and $(y^{(i)})_i$.

See proof in Section J.

Theorem 7 (Malicious-Abort QFactory is correct and secure). *Assuming Conjecture 1, and by ensuring that the probability of the family $\mathcal{F}$ to have two preimages for any image is bigger than a constant $p_a > 1/2$, then there exists a set of parameters p_c, t_c and n_c such that Protocol 7.1 is correct with probability exponentially close to 1 and basis-blind, i.e. for any QPT adversary $\mathcal{A}$:*

$$\Pr\left[\tilde{B}_1 = B_1 \mid ((B_1, B_2), \tilde{B}_1) \leftarrow (\pi_{A_4 \oplus} \| \mathcal{A})\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

More precisely, we need $t_c \in (1/2, p_c)$ to be a constant, and both t_c and n_c need to be polynomial in n and $\Omega(n)$.

See proof in Section J.

8 Verifiable QFactory

In the previous protocols, Malicious 4-states QFactory and Malicious 8-states QFactory, the produced qubits came with the guarantee of *basis-blindness* (Definition 9 and Definition 10). While this property refers to the ability of a malicious

adversary to guess the honest basis bit(s), it tells nothing about the actual state that a deviating server might produce. For a number of applications, and most notably for *verifiable blind quantum computation* [FK17], the basis-blindness property is not sufficient. What is required is a stronger property, *verification*, that ensures that the produced state was essentially prepared correctly, even in a malicious run.

8.1 Verifiable QFactory Functionality

There are two issues with trying to define a verification property for QFactory. The first is that the adversarial server can always abort, therefore the verification property can only ensure that the probability of non-aborting *and* cheat is negligible. The second issue is that, since the final state is in the hands of the server, the server can always apply a final deviation on the state[10]. This is not different from what happens in protocols that do have quantum communication. In that case, the adversarial receiver (server) can also apply a deviation on the state received before using the state in any subsequent protocol. This deviation could even be the server replacing the received state with a totally different state.

Here, we define the strongest notion of verifiable QFactory possible, which exactly captures the idea of being able to recover the ideal state from the real state without any knowledge of the (secret) index of the ideal state. In Section E we show that this notion is sufficient for any protocol that includes communication of random secret qubits of the form $\left|+_{L\pi/4}\right\rangle$ which includes a verifiable quantum computation protocol. Furthermore, in Section F we show that it is possible to relax slightly the definition of verifiable QFactory.

Definition 12 (Verifiable QFactory). *Consider a party that is given a state uniformly chosen from a set of eight states $S = \{\rho_L \,|\, L \in \{0,1,...,7\}\}$ or an abort bit, where S is basis-blind i.e. given a state sampled uniformly at random from S, it is impossible to guess the last two bits of the index L of the state within the set S with non-negligible advantage. We say that this party has a Verifiable QFactory if, it aborts with small probability and when he does not abort, there exists an isometry Φ, that is independent of the index L, such that:*

$$\Phi(\rho_L) \overset{\epsilon}{\approx} \left|+_{L\pi/4}\right\rangle \left\langle+_{L\pi/4}\right| \otimes \sigma_{junk} \tag{8.1}$$

where the state σ_{junk} is independent of the index L.

It is worth stressing, that if the security setting is computational (as in this work), the basis-blindness and the approximate equality above involve a QPT distinguisher, while the isometry Φ needs to be computable in polynomial time.

[10] However that deviation needs to be independent of anything that is secret.

8.2 Blind Self-Testing

Before giving a verifiable QFactory protocol, we define a new concept of blind self-testing, that will be essential in proving the security of the former. Self-testing is a technique developed [MY03, MMMO06, vDMMS07, MHYS12, Mck14] that ensures that given some measurement statistics, classical parties can be certain that some untrusted quantum state (and operations), that two or more quantum parties share, is essentially (up to some isometry) the state that the parties believe they have. In high-level, we are going to use a test of this kind in order to certify that the output of Verifiable QFactory is indeed the desired one.

Existing results, that we will call *non-local self-testing*, only deal with how to exploit the non-locality (the fact that the quantum state tested is shared between non-communicating parties) to test the state and operations. Naturally, the correctness is up to a local isometry (something that the servers can apply, while preserving the non-communication condition).

Here, instead of testing a single non-local state, we test a family of states, where the *non-locality* property is replaced by the *blindness* property - the fact that server is not aware (is blind) of which state from the possibly known family of states he is actually given in each run of the protocol. To see how this is closely related, one can imagine the usual non-local self-testing of the singlet state, where one quantum side (Alice) actually performs a measurement (as instructed). From the point of view of the other quantum side (Bob), he has a single state that, in the honest run, is one of the BB84 states, while he is totally oblivious about the basis of this state (if that was not the case, it would lead to signalling the basis choice of Alice's measurement). However, this is, by no means the most general case. Here we introduce the concept of *blind self-testing* formalising the above intuition.

We give here the most general case of blind self-testing and we conjecture that it holds. In Section G we list three simpler scenarios (of increasing complication) that lead to the most general case given here, following similar steps with the extension of simple i.i.d. self-testing to fully robust and rigid self-testing in existing literature [MMMO06, MHYS12, HYN12, RUV13, CGJV17]. In Appendix H, we provide the security proof of the first case, while full analysis of the most general blind self-testing goes beyond the scope of this work.

Protocol 8.1 Blind self-testing: The general case

– Server prepares a single state ρ_{tot} (consisting of N qubits in the honest run). This state has a corresponding index consisting of N 3-bit indices L_i ($L_i \in \{0, ..., 7\} \forall i \in \{1, ..., N\}$), and the server is basis blind with respect to each of these N indices, i.e. (being computationally bounded) he cannot determine with non-negligible advantage the two basis bits of any index L_i. On the other hand, client knows the indices L_i's.
– The client, randomly chooses a fraction f of the qubits to be used as tests and announces the set of corresponding indices $T = \{i_1, \cdots, i_{fN}\} \subset \{1, 2, ..., N\}$ to the server.
– For each test qubit $i_j \in T$, the client chooses a random measurement index $M_{i_j} \in \{000, \cdots, 111\}$ and instructs the server to measure the corresponding qubit in the $\left\{\left|+_{M_{i_j}\pi/4}\right\rangle, \left|-_{M_{i_j}\pi/4}\right\rangle\right\}$ basis.
– The server returns the test measurement results $\{c_{(i_j)}\}$.

— For each fixed pair (L, M), the client gathers all the test positions that correspond to that pair and from the relative frequencies, the client obtains an estimate for the probability $p_{L,M}$ (where by convention we have that $p_{L,M}$ corresponds to the $+1$ outcome, while $1 - p_{L,M}$ to the -1).
– If $|p_{L,M} - \cos^2((L-M)\pi/8)| \geq \epsilon_2$ for any pair (L, M) the client aborts.
Output: If the client does not abort (and this happens with non-negligible probability), then there exists an index-independent polynomial isometry $\Phi = \Phi_{k_1} \otimes \cdots \Phi_{k_l}$, given by products of the isometries in Fig. 5, that is applied to a random subset of non-tested qubits i, such that:

$$\Phi(Tr_{\text{all but } k_1,\cdots,k_l \text{ qubits}}\rho_{tot}) \overset{\epsilon(\epsilon_1,\epsilon_2)}{\approx} \left(\left|+_{L_{k_1}\pi/4}\right\rangle_{k_1} \otimes \cdots \otimes \left|+_{L_{k_l}\pi/4}\right\rangle_{k_l}\right) \otimes \sigma_{\text{junk}} \quad (8.2)$$

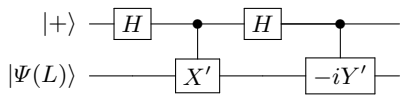

Fig. 5. The isometry of the blind self-testing. Note that the controlled gates are controlled in the X-basis, i.e. $\wedge U_{12}(a\,|+\rangle + b\,|-\rangle)_1 \otimes (|\psi\rangle)_2 = a\,|+\rangle_1 \otimes |\psi\rangle_2 + b\,|-\rangle_1 \otimes U\,|\psi\rangle_2$.

In this most general setting, we make no assumption on the state ρ_{tot} produced by server and want to recover the full tensor product structure of the resulting states given by Eq. (8.2). In the self-testing literature, Azuma-Hoeffding, quantum de-Finetti theorems and rigidity results [Hoe63, AZU67, CFS02, BH13] were used to uplift the simple i.i.d. case and prove security in the general setting.

8.3 The Verifiable QFactory Protocol

In this section we introduce a protocol for the final version of our functionality, Verifiable QFactory. Here, we give the protocol, show the correctness and the security, namely that the protocol achieves the verification property from Definition 12, based on the conjectured security of the most general *blind self-testing* given in Protocol 8.1. The basic idea is the following: repeat the Malicious 8-states QFactory multiple times, then the client chooses a random fraction of the output qubits and uses them for a test and next instructs the server to measure the test qubits in random angles and, finally, the client checks their statistics. Since the server does not know the states (or to be more precise, the basis bits), he is unlikely to succeed in guessing the correct statistics unless he is honest. (up to some trivial relabelling). The output qubits and the measurement angles, need to be from the set of 8-states, which is one of the reasons we wanted to give the 8-states extension of our Malicious 4-states QFactory. [11].

[11] This is actually related with Bell's theorem as one can see later from the similarity with self-testing results.

Protocol 8.2 Verifiable QFactory

Requirements: Same as in Protocol Protocol 3.1

Input: Client runs N times the algorithm $(k^{(i)}, t_k^{(i)}) \leftarrow \text{Gen}_{\mathcal{F}}(1^n)$, where $i \in \{1, \cdots, N\}$ denotes the ith run. He keeps the $t_k^{(i)}$'s private.

Protocol:

– Client: runs N times the Malicious 8-states QFactory Protocol 5.1.

– Client: records measurement outcomes $y^{(i)}, b^{(i)}$ and computes and stores the corresponding index of the output state $L^{(i)}$.

– Client: instructs the server to measure a random fraction rf of the output states, each in a randomly chosen basis of the form $\{\left|+_{M^{(i)}\pi/4}\right\rangle, \left|-_{M^{(i)}\pi/4}\right\rangle\}$. Here $M^{(i)}$ is the index of the measurement instructed.

– Server: returns the measurement outcomes $c^{(i)}$.

– Client: for each pair (L, M) collects the results $c^{(j)}$ for all j's that have the specific pair and with the relative frequency obtains an estimate for the probability $p(L, M)$.

– Client: aborts unless all the estimates of the probabilities $p(L, M)$ are ϵ-close to the ideal one i.e. $p(L, M) \overset{\epsilon}{\approx} |\left\langle +_{M\pi/4}\right| +_{L\pi/4}\rangle|^2$.

Output: Probability of non-abort and being far from the ideal state[12] is negligible ϵ':

$$p(\text{non-abort} \wedge \Delta(\rho_{L^{(i_1)}\cdots L^{(i_{N(1-f)})}}, \rho_{\text{ideal}}) \geq t(n)) \leq \epsilon' \tag{8.3}$$

where $i_1, ..., i_{N(1-f)}$ refer to the unmeasured qubits and where

$$\Phi(\rho_{\text{ideal}}) = \otimes_{k=1}^{N(1-f)} \left|+_{L^{(i_k)}\pi/4}\right\rangle \left\langle +_{L^{(i_k)}\pi/4}\right| \otimes \sigma_{\text{junk}} \tag{8.4}$$

and σ_{junk} is a constant density matrix, ϵ, ϵ' are negligible functions and $t(\cdot)$ is a non-negligible function.

Moreover, in an honest run, the probability of aborting is negligible and the output is:

$$\rho_{\text{honest}} = \otimes_{k=1}^{N(1-f)} \left|+_{L^{(i_k)}\pi/4}\right\rangle \left\langle +_{L^{(i_k)}\pi/4}\right| \tag{8.5}$$

Theorem 8 (correctness). *If Protocol 8.2 is run honestly, it aborts with negligible probability and the output (non-measured) qubits are exactly in a product state of the form* $\left|+_{L^{(i_k)}\pi/4}\right\rangle \left\langle +_{L^{(i_k)}\pi/4}\right|$. *Therefore, the trivial isometry (the identity) suffices to recover the state of Eq. (8.1), and where there is no junk state.*

Proof. In an honest run, each of the outputs of different Malicious 8-states QFactory runs, are of the correct form, therefore measuring any of those outputs in the $\{\left|\pm_{M\pi/4}\right\rangle\}$ basis returns the correct statistics with high probability. Hence, the protocol does not abort, while the remaining states are also prepared correctly. □

[12] The distance Δ used here depends on the setting. In our case it is understood as a QPT distinguisher.

Theorem 9 (security). *Protocol 8.2 is a Verifiable QFactory (Definition 12), i.e. the probability of accepting the tests and having a state far from the ideal is negligible irrespective of the deviation of the adversary, assuming that the self-testing Protocol 8.1 is correct.*

Sketch of Proof. The outputs of the 8-states QFactory are basis blind, and satisfying the measurement statistics too, leading exactly to the requirements of the definition of general blind self-testing in Protocol 8.1. It follows that there exists an isometry such as that requested by Eq. (8.4).

□

See also in Section G and Section H, where we explain how this task is very similar with self-testing results, and provide the first step for our self-testing result.

9 Acknowledgements

LC is very grateful to Céline Chevalier for all the discussions he had with her, and to Antoine Joux for the very pertinent comments. He would also like to give a special thanks to Geoffroy Couteau, Omar Fawzi and Alain Passelgue who gave him great advices concerning security proof methods. AC and PW are very grateful to Atul Mantri, Thomas Zacharias, Yiannis Tselekounis and Vedran Dunjko for very helpful and interesting discussions. The work was supported by the following grants FA9550-17-1-0055, EPSRC grants: EP/N003829/1 and EP/M013243/1, and by the French ANR Project ANR-18-CE39-0015 CryptiQ.

References

ABEM17. Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive Proofs for Quantum Computations. *arXiv e-prints*, page arXiv:1704.04487, Apr 2017.

ACGK17. Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. On the implausibility of classical client blind quantum computing. *arXiv preprint arXiv:1704.08482*, 2017.

ADSS17. Gorjan Alagic, Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum fully homomorphic encryption with verification. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 438–467. Springer, 2017.

AGKP14. Frederik Armknecht, Tommaso Gagliardoni, Stefan Katzenbeisser, and Andreas Peter. General impossibility of group homomorphic encryption in the quantum world. In *International Workshop on Public Key Cryptography*, pages 556–573. Springer, 2014.

AS03. Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 04, 10 2003.

AZU67. KAZUOKI AZUMA. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

BCM+18. Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 320–331, 2018.

BFK09. Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 517–526, Washington, DC, USA, 2009. IEEE Computer Society.

BH13. Fernando G.S.L. Brandao and Aram W. Harrow. Quantum de finetti theorems under local measurements with applications. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 861–870, New York, NY, USA, 2013. ACM.

BJ15. Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low t-gate complexity. In *Annual Cryptology Conference*, pages 609–629. Springer, 2015.

Bra18. Zvika Brakerski. Quantum FHE (almost) as secure as classical. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 67–95. Springer, 2018.

Can00. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://eprint.iacr.org/2000/067`.

CCKW18. Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. On the possibility of classical client blind quantum computing. *CoRR*, abs/1802.08759, 2018.

CFS02. Carlton M. Caves, Christopher A. Fuchs, and Rdiger Schack. Unknown quantum states: The quantum de finetti representation. *Journal of Mathematical Physics*, 43(9):4537–4559, 2002.

CGJV17. Andrea Coladangelo, Alex Grilo, Stacey Jeffery, and Thomas Vidick. Verifier-on-a-leash: new schemes for verifiable delegated quantum computation, with quasilinear resources. *arXiv preprint arXiv:1708.07359*, 2017.

Chi05. Andrew M. Childs. Secure assisted quantum computation. *Quantum Info. Comput.*, 5(6):456–466, September 2005.

DK16. Vedran Dunjko and Elham Kashefi. Blind quantum computing with two almost identical states. *arXiv e-prints*, page arXiv:1604.01586, April 2016.

DKL11. Vedran Dunjko, Elham Kashefi, and Anthony Leverrier. Blind quantum computing with weak coherent pulses. *Physical Review Letters*, 108:200502, 08 2011.

DSS16. Yfke Dulek, Christian Schaffner, and Florian Speelman. Quantum homomorphic encryption for polynomial-sized circuits. In *Annual Cryptology Conference*, pages 3–32. Springer, 2016.

FHM18. Joseph F. Fitzsimons, Michal Hajdusek, 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. *Phys. Rev. A*, 98:022323, Aug 2018.

GMMR13. Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G. Rudolph. Efficient universal blind quantum computation. *Phys. Rev. Lett.*, 111:230501, Dec 2013.

GNW11. Oded Goldreich, Noam Nisan, and Avi Wigderson. *On Yao's XOR-Lemma*, pages 273–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

GV19. Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. *arXiv e-prints*, page arXiv:1904.06320, Apr 2019.

Hoe63. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

HYN12. Tzyh Haur Yang and Miguel Navascus. Robust self testing of unknown quantum systems into any entangled two-qubit states. *Physical Review A*, 87, 10 2012.

IALL89. Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudorandom generation from one-way functions. pages 12–24, 01 1989.

KSd04. Hartmut Klauck, Robert Spalek, and Ronald de Wolf. Quantum and Classical Strong Direct Product Theorems and Optimal Time-Space Tradeoffs. *arXiv e-prints*, pages quant–ph/0402123, February 2004.

LC17. Ching-Yi Lai and Kai-Min Chung. On statistically-secure quantum homomorphic encryption. *arXiv preprint arXiv:1705.00139*, 2017.

Lia15. Min Liang. Quantum fully homomorphic encryption scheme based on universal quantum circuit. *Quantum Information Processing*, 14(8):2749–2759, 2015.

Mah18a. Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In *FOCS*, pages 332–338. IEEE Computer Society, 2018.

Mah18b. Urmila Mahadev. Classical verification of quantum computations. In *FOCS*, pages 259–267. IEEE Computer Society, 2018.

Mck14. Matthew Mckague. Self-testing graph states. In *Revised Selected Papers of the 6th Conference on Theory of Quantum Computation, Communication, and Cryptography - Volume 6745*, TQC 2011, pages 104–120, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

MDK15. Tomoyuki Morimae, Vedran Dunjko, and Elham Kashefi. Ground state blind quantum computation on aklt state. *Quantum Info. Comput.*, 15(3-4):200–234, March 2015.

MF12. Tomoyuki Morimae and Keisuke Fujii. Blind topological measurement-based quantum computation. *Nature communications*, 3:1036, 09 2012.

MHYS12. Matthew Mckague, Tzyh Haur Yang, and Valerio Scarani. Robust self testing of the singlet. *Journal of Physics A: Mathematical and Theoretical*, 45, 03 2012.

MMMO06. Frederic Magniez, Dominic Mayers, Michele Mosca, and H Ollivier. Self-testing of quantum circuits. 01 2006.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

MPDF13. Atul Mantri, Carlos A Pérez-Delgado, and Joseph F Fitzsimons. Optimal blind quantum computation. *Physical review letters*, 111(23):230502, 2013.

MR11. Ueli Maurer and Renato Renner. Abstract cryptography. In *IN INNOVATIONS IN COMPUTER SCIENCE*. Tsinghua University Press, 2011.

MY03. Dominic Mayers and Andrew Yao. Self testing quantum apparatus. *Quantum information & computation*, 4:273, 08 2003.

NC10. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

NS17. Michael Newman and Yaoyun Shi. Limitations on transversal computation through quantum homomorphic encryption. *arXiv preprint arXiv:1704.07798*, 2017.

OTF15. Yingkai Ouyang, Si-Hui Tan, and Joseph Fitzsimons. Quantum homomorphic encryption from quantum codes. *arXiv preprint arXiv:1508.00938*, 2015.

Pei09. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

Pre18. John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

PW07. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. Cryptology ePrint Archive, Report 2007/279, 2007. `https://eprint.iacr.org/2007/279`.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

RUV13. Ben W. Reichardt, Falk Unger, and Umesh Vazirani. A classical leash for a quantum system: Command of quantum systems via rigidity of chsh games. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 321–322, New York, NY, USA, 2013. ACM.

She10. Alexander A. Sherstov. Strong direct product theorems for quantum communication and query complexity. *arXiv e-prints*, page arXiv:1011.4935, November 2010.

TKO+16. Si-Hui Tan, Joshua A Kettlewell, Yingkai Ouyang, Lin Chen, and Joseph F Fitzsimons. A quantum approach to homomorphic encryption. *Scientific reports*, 6:33467, 2016.

TSSR10. Marco Tomamichel, Christian Schaffner, Adam Smith, and Renato Renner. Leftover Hashing Against Quantum Side Information. *arXiv e-prints*, page arXiv:1002.2436, Feb 2010.

vDMMS07. W. van Dam, F. Magniez, M. Mosca, and M. Santha. Self-testing of universal and fault-tolerant sets of quantum gates. *SIAM Journal on Computing*, 37(2):611–629, 2007.

VW08. Emanuele Viola and Avi Wigderson. Norms, xor lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4(7):137–168, 2008.

WEH18. Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412):303, 2018.

YPDF14. Li Yu, Carlos A Pérez-Delgado, and Joseph F Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Physical Review A*, 90(5):050303, 2014.

Supplementary Material

A Function Construction proofs

A.1 Proof of Theorem 4, homomorphicity

To prove that g_K is homomorphic, we notice that:

$$\begin{aligned} g_K(s_1, e_1, d_1) + g_K(s_2, e_2, d_2) &= \bar{g}_K(s_1, e_1) + d_1 \cdot v + \bar{g}_K(s_2, e_2) + d_2 \cdot v \bmod q \\ &= \bar{g}_K(s_1 + s_2 \bmod q, e_1 + e_2) + (d_1 + d_2) \cdot v \bmod q \\ &= g_K(s_1 + s_2 \bmod q, e_1 + e_2, d_1 \oplus d_2) \end{aligned}$$

where for the last equality we used the fact that if $d_1, d_2 \in \{0, 1\}$, then $d_1 \cdot \frac{q}{2} + d_2 \cdot \frac{q}{2} \bmod q = (d_1 \oplus d_2) \cdot \frac{q}{2} \bmod q$.

We make the following remark: the proof is constructed for the case when $\bar{g}$ is perfectly homomorphic, but it also holds in the case when $\bar{g}$ is homomorphic with high probability, resulting in g being homomorphic with the same high probability.

A.2 Proof of Theorem 4, one-wayness

To prove the **one-wayness** of g, we are going to reduce it to the one-wayness of $\bar{g}$.

Thus, we assume there exists a QPT adversary $\mathcal{A}$ that can invert g with probability P and we construct a QPT adversary $\mathcal{A}'$ inverting $\bar{g}$ with the same probability P.

$Invert_{\mathcal{A}',K}(y)$

1 : $d \leftarrow_{\$} \{0, 1\}$
2 : $y' \leftarrow y + d \cdot v$
3 : $(s', e', d') \leftarrow \mathcal{A}_K(y')$
4 : **return** (s', e')

A.3 Proof of Theorem 4, injectivity

To prove this, we will use the injectivity property of the function $\bar{g}$.

$$g_K(s, e, d) = \bar{g}_K(s, e) + d \cdot v$$

Assume there exist 2 tuples (s_1, e_1, d_1) and (s_2, e_2, d_2) such that $g(s_1, e_1, d_1) = g(s_2, e_2, d_2)$. To prove that g is injective we must show that $(s_1, e_1, d_1) = (s_2, e_2, d_2)$. This is equivalent to:

$$\begin{aligned} \bar{g}_K(s_1, e_1) + d_1 \cdot v &= \bar{g}_K(s_2, e_2) + d_2 \cdot v \\ \bar{g}_K(s_1, e_1) - \bar{g}_K(s_2, e_2) + (d_1 - d_2) \cdot v &= 0 \end{aligned} \tag{A.1}$$

Now, if $d_1 = d_2$, then we have that $\bar{g}(s_1, e_1) - \bar{g}(s_2, e_2) = 0$, and because $\bar{g}$ is injective, this would imply that $s_1 = s_2$ and $e_1 = e_2$ and thus g would also be injective.
Let us suppose that $d_1 \neq d_2$ and we will prove that this is impossible. Without loss of generality we can assume that $d_1 = 0$ and $d_2 = 1$.

Thus, we want to show that it is impossible to have (s_1, e_1) and (s_2, e_2) such that:

$$\bar{g}_K(s_1, e_1) - \bar{g}_K(s_2, e_2) = v \tag{A.2}$$

Equation is equivalent to:

$$Ks_1 + e_1 - Ks_2 - e_2 = \begin{pmatrix} \frac{q}{2} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \bmod q \tag{A.3}$$

This can be rewritten as:

$$K_{1,1}(s_{1,1} - s_{2,1}) + ... + K_{1,n}(s_{1,n} - s_{2,n}) + (e_{1,1} - e_{2,1}) = \frac{q}{2} \bmod q \tag{A.4}$$

$$K_{i,1}(s_{1,1} - s_{2,1}) + ... + K_{i,n}(s_{1,n} - s_{2,n}) + (e_{1,i} - e_{2,i}) = 0 \bmod q \tag{A.5}$$

$\forall\, i = 2, ..., m$ and where $s_{1,i}$ and $s_{2,i}$ are the i-th elements of s_1 and s_2 respectively and $e_{1,i}$ and $e_{2,i}$ are the i-th elements of e_1, respectively e_2.

Now, as the function $\bar{g}_K : \mathbb{Z}_q^n \times E^m \to \mathbb{Z}_q^m$, where $K \leftarrow \mathbb{Z}_q^{m \times n}$ is injective, the following function is also injective:
$\bar{g_1}_{K_1} : \mathbb{Z}_q^n \times E^{m-1} \to \mathbb{Z}_q^{m-1}$, where $K_1 \leftarrow \mathbb{Z}_q^{m-1 \times n}$ and where $\bar{g}$ and $\bar{g_1}$ have the exact same definition:

$$\bar{g_1}_{K_1}(s, e) = K_1 s + e \bmod q$$

More specifically, the only difference between the 2 functions is the change of dimension from m to $m - 1$, but as the injectivity proof from [MP12] holds for any $m = \Omega(n)$, then $\bar{g_1}$ is also injective (if we want an exact argument, it is also possible to imagine that instead of taking K directly, we add one more line on top of K and e, use the lower part to recover s_0, and then use classic algebra to recover d).

Now, consider the matrix K_1 obtained from K by removing the first line. As shown above, $\bar{g_1}_{K_1}$ is an injective function, thus from Equation A.5, we get that:

$$s_1 = s_2 \tag{A.6}$$

$$e_{1,i} = e_{2,i}\ \forall\, i = 2, .., m \tag{A.7}$$

Now as $s_1 = s_2$, from Equation A.4, we obtain $e_{1,1} - e_{2,1} = \frac{q}{2}$.
However, from the domain of $\bar{g}$, we have that: $|e_{1,1} - e_{2,1}| < 2\mu < \frac{q}{2}$ (where for the last inequality we used Lemma 3). Contradiction.

A.4 Proof of Lemma 2

To prove that h is a hardcore predicate of g_K, we must prove that for any QPT adversary $\mathcal{A}$, we have:

$$\Pr_{\substack{s \leftarrow\$ \mathbb{Z}_q^n \\ e \leftarrow\$ E^m \\ d \leftarrow\$ \{0,1\}}} [\mathcal{A}(1^\lambda, K, g_K(s,e,d)) = h(s,e,d)] \leq \tfrac{1}{2} + \mathsf{negl}(\lambda) \tag{A.8}$$

Using the definitions of the 2 functions, we can express it as:

$$\Pr_{\substack{s \leftarrow\$ \mathbb{Z}_q^n \\ e \leftarrow\$ E^m \\ d \leftarrow\$ \{0,1\}}} [\mathcal{A}(1^\lambda, K, Ks + e + d \cdot v) = d] \leq \tfrac{1}{2} + \mathsf{negl}(\lambda) \tag{A.9}$$

This is equivalent to prove that the distributions $D_1 = \{K, Ks + e\}$ and $D_2 = \{K, Ks + e + v\}$ are indistinguishable[13], or equivalently:

$$\{K_i, \langle K_i, s\rangle + e_i\}_{i=1}^m \overset{c}{\approx} \{K_i, \langle K_i, s\rangle + e_i + v_i\}_{i=1}^m \tag{A.10}$$

Using the decisional LWE assumption we already know that when u_i are uniform chosen from $\mathbb{Z}_q$, we have [14]:

$$\{K_i, \langle K_i, s\rangle + e_i\}_{i=1}^m \overset{c}{\approx} \{K_i, u_i\}_{i=1}^m \tag{A.11}$$

Then, as v is a fixed vector, we also have that:

$$\{K_i, \langle K_i, s\rangle + e_i + v_i\}_{i=1}^m \overset{c}{\approx} \{K_i, u_i\}_{i=1}^m \overset{c}{\approx} \{K_i, \langle K_i, s\rangle + e_i\}_{i=1}^m \tag{A.12}$$

which completes the proof.

Note also that we can easily notice the homomorphicity property of the defined function h:

$$\begin{aligned} h(s_1, e_1, d_1) \oplus h(s_2, e_2, d_2) &= d_1 \oplus d_2 \\ = h(s_1 + s_2 \bmod q, e_1 + e_2 \bmod q, d_1 \oplus d_2) & \end{aligned} \tag{A.13}$$

B Malicious 8-states QFactory - Proof of Theorem 6

Proof. We prove the following:
If there exists a QPT adversary $\mathcal{A}$ that is able to break the 8-states basis blindness property of Malicious 8-states QFactory (determine the indices L_2 and L_3 with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$ for some polynomial function $poly_1$), then we can

[13] It is also easy to write an explicit reduction

[14] this holds because the parameters (fully given in [CCKW18, Lemma 6.9]) of the function are chosen to make y_0 indistinguishable from a random vector by a direct reduction to LWE

construct a QPT adversary $\mathcal{A}'$ that can break the 4-states basis blindness of the Malicious 4-states QFactory protocol (determine the basis bit with probability $\frac{1}{2} + \frac{1}{poly_2(n)}$, for some polynomial $poly_2(\cdot)$).

The input to $\mathcal{A}'$ should be consisting only of the $\mathcal{F}$ family index, k, and the description of h. Next we show how to construct $\mathcal{A}'$ to determine the corresponding index B_1 or B_1' of the output state (of one of the 2 Malicious 4-states QFactory runs), by using as a subroutine $\mathcal{A}$ that acts as follows: receives as input 2 function indices $k^{(1)}$ and $k^{(2)}$, runs Malicious 8-states QFactory and then $\mathcal{A}$ is able to output the correct basis bits L_2 and L_3, with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$.

Before we describe $\mathcal{A}'$, we need to define the following 3 values:
P_2 = probability that $\mathcal{A}$ guesses correctly L_2;
P_3 = probability that $\mathcal{A}$ guesses correctly L_3;
$P_\oplus$ = probability that $\mathcal{A}$ guesses correctly $L_2 \oplus L_3$;

Now, given that $\mathcal{A}$ is able to produce both L_2 and L_3 with probability $\frac{1}{4} + \frac{1}{poly_1(n)}$, this implies that $max(P_2, P_3, P_\oplus) \geq \frac{1}{2} + \frac{1}{poly_2(n)}$ for some polynomial $poly_2(\cdot)$ (see proof in Section C).
We will construct $\mathcal{A}'$ such that if P_3 is the maximum, then $\mathcal{A}'$ can determine B_1 (break the basis blidness of the first Malicious 4-states QFactory run) and if P_2 or $P_\oplus$ is the maximum, then $\mathcal{A}'$ can determine B_1' (break the basis blidness of the the second "rotated" Malicious 4-states QFactory run). You can find the reduction in next page:

$\mathcal{A}'(k, h, 1^n)$

1 : // $k^{(1)}$ corresponds to the input for the first run of Malicious 4-states QFactory
2 : // - with the output index (B_1, B_2), while k corresponds to
3 : // the input for the second run - with the output index (B_1', B_2')
4 : $(k^{(1)}, t_k^{(1)}) \leftarrow_\$ Gen_{\mathcal{F}}(1^n)$
5 : // As the probability P of successfully guessing L_2 and L_3 is $\frac{1}{4} + \frac{1}{poly_1(n)}$
6 : // We know that $max(P_2, P_3, P_\oplus) \geq \frac{1}{2} + \frac{1}{poly_2(n)}$
7 : **if** $P_3 = max(P_2, P_3, P_\oplus)$
8 : // we break the basis-blindness of the first Malicious 4-states QFactory by determining B_1
9 : $(y^{(1)}, y^{(2)}, b^{(1)}, b^{(2)}, s_1, s_2), (\tilde{L_2}, \tilde{L_3}) \leftarrow \mathcal{A}(k, k^{(1)}, h)$
10 : // $(y^{(1)}, y^{(2)}, b^{(1)}, b^{(2)}, s_1, s_2)$ represents the classical communication received from $\mathcal{A}$
11 : // during the run of Malicious 8-states QFactory, and
12 : // $(\tilde{L_2}, \tilde{L_3})$ - are the guesses of $\mathcal{A}$ for the indices of the outcome
13 : $\tilde{B_1} \leftarrow \tilde{L_3}$
14 : **return** $\tilde{B_1}$// as $B_1 = 1 \oplus L_3$ as seen in Eq. 5.9 and
15 : // we have success probability $\geq \frac{1}{2} + \frac{1}{poly_2(n)}$
16 : **else**
17 : // we break the basis-blindness of the second Malicious 4-states QFactory by determining B_1'
18 : $(y^{(1)}, y^{(2)}, b^{(1)}, b^{(2)}, s_1, s_2), (\tilde{L_2}, \tilde{L_3}) \leftarrow \mathcal{A}(k^{(1)}, k, h)$
19 : // $(y^{(1)}, y^{(2)}, b^{(1)}, b^{(2)}, s_1, s_2)$ represents the classical communication received from $\mathcal{A}$
20 : // during the run of Malicious 8-states QFactory, and
21 : // $(\tilde{L_2}, \tilde{L_3})$ - are the guesses of $\mathcal{A}$ for the indices of the outcome
22 : $(z^{(1)}, z'^{(1)}) \leftarrow Inv_{\mathcal{F}}(y^{(1)}, t_k^{(1)})$
23 : $B_1 \leftarrow h(z^{(1)}) \oplus h(z'^{(1)})$
24 : $B_2 \leftarrow [b_n^{(2)} + \sum(z_i^{(2)} \oplus z_i'^{(2)}) \cdot b_i^{(2)}]B_1 + h(z^{(2)})(1 - B_1) \bmod 2$
25 : **if** $P_2 = max(P_2, P_3, P_\oplus)$
26 : // Then $B_1' = L_2 \oplus B_1 \cdot (B_2 \oplus s_2)$ as seen in Eq. 5.8
27 : $\tilde{B_1'} \leftarrow \tilde{L_2} \oplus [B_1 \cdot (B_2 \oplus s_2)]$
28 : **return** $\tilde{B_1'}$
29 : **if** $P_\oplus = max(P_2, P_3, P_\oplus)$
30 : $\tilde{B_1'} \leftarrow \tilde{L_2} \oplus \tilde{L_3} \oplus B_1 \oplus [B_1 \cdot (B_2 \oplus s_2)]$
31 : **return** $\tilde{B_1'}$

□

C Probability of guessing two predicates

Lemma 9 (Implication of guessing two predicates).
Let $(a,b) \in \{0,1\}^2$ be two bits sampled uniformly at random. Let f be any function of (a,b) (eventually randomized). Then if $\mathcal{A}$ is an adversary such that $\Pr[\mathcal{A}(f(a,b)) = (a,b)] \geq 1/4 + \frac{1}{\mathsf{poly}(n)}$ (where the probability is taken over the choice of a and b, the randomness of f and $\mathcal{A}$), then either:

- *$\mathcal{A}$ is good to guess a, i.e.:*
  $P_1 = \Pr\left[\tilde{a} = a \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq 1/2 + 1/\mathsf{poly}(n)$
- *$\mathcal{A}$ is good to guess b, i.e.:*
  $P_2 = \Pr\left[\tilde{b} = b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq 1/2 + 1/\mathsf{poly}(n)$
- *$\mathcal{A}$ is good to guess the XOR of a and b, i.e.:*
  $P_\oplus = \Pr\left[\tilde{a} \oplus \tilde{b} = a \oplus b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq 1/2 + 1/\mathsf{poly}(n)$

Proof. Let's denote by:

- $e_1 = \Pr\left[\tilde{a} \neq a \text{ and } \tilde{b} \neq b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right]$
- $e_2 = \Pr\left[\tilde{a} = a \text{ and } \tilde{b} \neq b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right]$
- $e_3 = \Pr\left[\tilde{a} \neq a \text{ and } \tilde{b} = b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right]$
- $e_4 = \Pr\left[\tilde{a} = a \text{ and } \tilde{b} = b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right]$

Now, let us assume that the probability to do a correct guess is good, i.e. $e_4 \geq \frac{1}{4} + \frac{1}{\mathsf{poly}(n)}$. Because the probability of guessing correctly a (resp b) is bad, we have $e_2 + e_4 \leq \frac{1}{2} + \mathsf{negl}(n)$ (resp. $e_3 + e_4 \leq \frac{1}{2} + \mathsf{negl}(n)$), so $e_2 \leq \frac{1}{4} + \mathsf{negl}(n)$ (resp. $e_3 \leq \frac{1}{4} + \mathsf{negl}(n)$). So $e_2 + e_3 \leq \frac{1}{2} - \frac{1}{\mathsf{poly}(n)}$, and because $e_1 + e_2 + e_3 + e_4 = 1$, we get $e_1 + e_4 \geq \frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$. But $e_1 + e_4$ is exactly the probability to guess the XOR, i.e.

$$\Pr\left[\tilde{a} \oplus \tilde{b} = a \oplus b \mid (\tilde{a},\tilde{b}) \leftarrow \mathcal{A}(f(a,b))\right] \geq \frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$$

□

D Proofs for blind measurement gadget Figure 3

Assume the server has a state $|\Psi\rangle_{AE}$, where A is single qubit Hilbert space. We say that the server performed a Z measurement if with probability $p_{Z,\Psi}(m)$ he obtains the outcome m and his remaining (non-measured) state is at $|\Psi_m^Z\rangle_E$. Similarly for X measurement, we have $p_{X,\Psi}(m)$ and $|\Psi_m^X\rangle_E$. Note, that the state $|\Psi\rangle_{AE}$ can be written in either basis: $\sum_m \psi_m^X |m\rangle_A |\Psi_m^X\rangle_E$ or $\sum_m \psi_m^Z |m\rangle_A |\Psi_m^Z\rangle_E$, where $|\psi_m^X|^2 = p_{X,\Psi}(m)$ and $|\psi_m^Z|^2 = p_{Z,\Psi}(m)$.

As notation, let us call $p_u(m) = 1/2$ the unbiased distribution.

Lemma 10 (Correctness). *Following Figure 3 results in measuring the first qubit of the state $|\Psi\rangle_{AE}$ in the X-basis when $B_1 = 0$ and in the Z-basis when $B_1 = 1$.*

Proof. To check that Figure 3 does the desired functionality (in the honest run), we check separately the cases for different values of B_1.

Case $B_1 = 0$: The output state is $|s_1\rangle |s_2\rangle \left|\Psi^X_{s_2 \oplus B_2}\right\rangle_E$, while the probability to obtain $p(s_2) = p_{X,\Psi}(s_2 \oplus B_2)$ and s_1 occurs with probability $1/2 = p_u(s_1)$.

Case $B_1 = 1$: The output state is $|s_1\rangle |s_2\rangle \left|\Psi^Z_{s_1 \oplus B_2}\right\rangle_E$, while the probability to obtain $p(s_1) = p_{Z,\Psi}(s_1 \oplus B_2)$ and s_2 occurs with probability $1/2 = p_u(s_2)$.

We can see that the remaining state is indeed the correct one (provided one "corrects" the measurement outcomes by adding the B_2 bit). Similarly the probabilities are the desired ones, with only subtlety, that the relevant measurement outcome varies. If the measurement performed was X it is s_2 that is the important measurement bit, while if the measurement was Z it is the bit s_1.

□

Lemma 11 (Security). *An adversarial server cannot find out the basis of the measurement, irrespective of the state that he wants to measure or his deviation, with any non-negligible probability.*

Sketch Proof. The key idea is that an adversary that could succeed in this task, could use this strategy to break the (assumed) blindness of the input, and works by contradiction constructing the corresponding reduction.

Consider a QPT adversary $\mathcal{A}$, that for some fixed state $|\Phi\rangle_{AE}$, participates at the given blind-measurement protocol (but can deviate as he wishes) and succeeds in guessing the basis of the measurement that is to be performed with non-negligible probability $\frac{1}{p(n)}$. Then we construct another QPT adversary $\mathcal{A}'$ that succeeds in breaking the security of the basic QFactory protocol with same (non-negligible) probability $\frac{1}{p(n)}$.

$\mathcal{A}'(k)$

1 : Run Malicious 4-states QFactory with input k and $\mathcal{A}$ as Server;

2 : $\mathcal{A}$ obtains $|in\rangle = H^{B_1} X^{B_2} |0\rangle$

3 : Apply Blind Measurement Gadget (Figure 3) on input $|in\rangle$ and $|\Phi\rangle_{AE}$

4 : **return** $\mathcal{A}(|\Phi\rangle_{AE})$// succeeds with prob $\frac{1}{p(n)}$

□

E Replacing a quantum channel with verifiable QFactory and Verifiable Quantum Computation

Here we prove that the verifiable QFactory can be used to replace *any* protocol that has a quantum channel where the honest parties send random secret qubits

of the form $\left|+_{L\pi/4}\right\rangle$. We show this, with a simple reduction: if there exist a QPT adversary $\mathcal{A}$ that can break the protocol with the verifiable QFactory states ρ_L with non-negligible probability $\frac{1}{p(n)}$, then there exist a QPT adversary $\mathcal{A}'$ that can break the security of the initial protocol that has quantum communication of the form $\left|+_{L\pi/4}\right\rangle$, with the same probability of success.

$\mathcal{A}'(\left|+_{L\pi/4}\right\rangle)$

1 : *Prepare* σ_{junk} // junk does not depend on θ

2 : $\gamma \leftarrow \left|+_{L\pi/4}\right\rangle\left\langle+_{L\pi/4}\right| \otimes \sigma_{\text{junk}}$

3 : $\gamma' \leftarrow \Phi^{-1}\left(\left|+_{L\pi/4}\right\rangle\left\langle+_{L\pi/4}\right| \otimes \sigma_{\text{junk}}\right) = \rho_L$ // Φ is a QPT isometry

4 : **return** $\mathcal{A}(\rho_L)$ // succeeds with prob $\frac{1}{p(n)}$

One of the most important possible applications of Verifiable QFactory is the classical client verifiable blind quantum computation. In particular, the verifiable blind quantum computation protocol of [FKD18] requires quantum communication in the beginning of the protocol and consists of strings of states of the form $\left|+_{L\pi/4}\right\rangle$ sent from the client to the server. According to our proof above, this means that the protocol of [FKD18] is as secure as a classical client protocol that replaces the quantum channel with verifiable QFactory.

F Strong Blindness

We prove here that for our purposes, it is possible to slightly relax Definition 12. In particular, given that the set of states $S = \{\rho_L \,|\, L \in \{0,1,...,7\}\}$ is *basis-blind*, and that there exists an index-independent isometry Φ that maps a state ρ_L to a state ϵ-close to $\left|+_{L\pi/4}\right\rangle\left\langle+_{L\pi/4}\right| \otimes \sigma(L)$, then we can *prove* that the junk state $\sigma(L)$ has to be independent of L and thus satisfies Definition 12.

This property is (trivially) related with what we call "*strong blindness*", since it essentially means that basis blindness (along with the existence of some isometry) guarantees that the only information that the server can learn from L is exactly the information he can learn in an honest run.

Definition 13 (8-states Strong Blindness). *Consider a party that is given a state uniformly chosen from a set of eight states $S = \{\rho_L \,|\, L \in \{0,1,...,7\}\}$. We say that S is strongly blind if the information the party can learn about the index L is bounded by the information that one can obtain from the state $\left|+_{L\pi/4}\right\rangle$.*

Lemma 12 (8-states strong blindness from isometry and weak blindness). *Consider a party that is given a state uniformly chosen from a set of eight states $\{\rho_L\}$, where the set is basis blind. Assume also that there exists an isometry Φ, that is independent of the index L, such that*

$$\Phi(\rho_L) \overset{\epsilon}{\approx} \left|+_{L\pi/4}\right\rangle\left\langle+_{L\pi/4}\right| \otimes \sigma(L) \tag{F.1}$$

where the state $\sigma(L)$ is a general junk state. Then we can show that $\sigma(L)$ should have negligible dependence on L, and therefore, by applying the inverse of the isometry, it satisfies 8-states strong blindness.

Proof. We prove this by contradiction. Assume that the state $\sigma(L)$ does not have negligible dependence on L. This means that one can guess L with non-negligible advantage from a random guess, i.e. there exists some measurement such that when applied to $\sigma(L)$ the measurement outcome is L with probability $Pr_L(\sigma(L)) = \frac{1}{8} + \frac{1}{poly(n)}$, for some polynomial *poly*.
From the 8 state basis blindness condition, we have that:

$$\sigma(L) + \sigma(L \oplus 100) \overset{\epsilon}{\approx} \sigma(L') + \sigma(L' \oplus 100) \ \forall \ L, L' \in \{0, 1, ..., 7\}, \tag{F.2}$$

where $\oplus$ refers to bitwise xor.
Then, from this condition, we also deduce that: $Pr_{L\oplus 100}(\sigma(L)) = \frac{1}{8} - \frac{1}{poly(n)}$.
Now, consider the 2 bases: $\{L, L \oplus 100\}$ and $\{L \oplus 010, L \oplus 110\}$. We will show next how to construct a distinguisher between these 2 basis. The idea is to use the information about the index L obtained from $\sigma(L)$ and then perform an optimal quantum measurement on the $|+_{L\pi/4}\rangle$ state given the prior knowledge that we obtained from $\sigma(L)$. This will lead to guessing information about the basis bit with non-negligible probability.

Without loss of generality, we suppose $Pr_{L\oplus 010} \geq Pr_{L\oplus 110}$.
Then we get:

$$\begin{aligned} Pr\left(L \, or \, L \oplus 010\right) = Pr_L + Pr_{L\oplus 010} = Pr_{L\oplus 100} + \frac{2}{poly(n)} + Pr_{L\oplus 010} \geq \\ Pr_{L\oplus 100} + Pr_{L\oplus 110} + \frac{2}{poly(n)} = Pr\left(L \oplus 100 \, or \, L \oplus 110\right) \end{aligned} \tag{F.3}$$

Then, to distinguish between the 2 bases, we will make a measurement on the $|+_{L\pi/4}\rangle$ state (first register), in the basis $\{L \oplus 011, L \oplus 111\}$. The resulting distinguishing probability is:

$$\begin{aligned} Pr_{success} &= \frac{2+\sqrt{2}}{4} \cdot \left(\frac{1}{2} + \frac{1}{poly(n)}\right) + \left(1 - \frac{2+\sqrt{2}}{4}\right) \cdot \left(\frac{1}{2} - \frac{1}{poly(n)}\right) \\ &= \frac{1}{2} + \frac{1}{poly'(n)} \end{aligned} \tag{F.4}$$

which, given the basis blindness assumption, reaches a contradiction. □

G Blind self-testing intermediate scenarios

Here we introduce a number of scenarios that takes us from a very simple setting for blind self-testing to the general case of Protocol 8.1.

Protocol G.1 Blind Self-Testing: The independent identically distributed case (Scenario 1)

– The server chooses eight states $\{|\Psi(000)\rangle, |\Psi(001)\rangle, \cdots, |\Psi(111)\rangle\}$, such that they are basis blind[15], i.e.

$$\Delta(\rho_L + \rho_{L\oplus 100}, \rho_{L'} + \rho_{L'\oplus 100}) \leq \epsilon_1 \quad \forall\ L, L' \in \{0, 1, ..., 7\} \tag{G.1}$$

where $\rho_L := |\Psi(L)\rangle \langle\Psi(L)| \ \ \forall L \in \{0, 1, ..., 7\}$.
– The client chooses randomly N indices, $\{L_1, \cdots, L_N\}$, where each $L_i \leftarrow \{000, \cdots, 111\}$ and sends the set of states $\{|\Psi(L_1)\rangle, |\Psi(L_2)\rangle, \cdots, |\Psi(L_N)\rangle\}$ to the server, while keeping the indices L_i secret.
– The client, randomly chooses a fraction f of the qubits to be used as tests and announces the set of corresponding indices $T = \{i_1, \cdots, i_{fN}\} \subset \{1, 2, ..., N\}$ to the server.
– For each test qubit $i_j \in T$, the client chooses a random measurement index $M_{i_j} \in \{000, \cdots, 111\}$ and instructs the server to measure the corresponding qubit in the $\left\{\left|+_{M_{i_j}\pi/4}\right\rangle, \left|-_{M_{i_j}\pi/4}\right\rangle\right\}$ basis.
– The server returns the test measurement results $\{c_{(i_j)}\}$.
– For each fixed pair (L, M), the client gathers all the test positions that correspond to that pair and from the relative frequencies, the client obtains an estimate for the probability $p_{L,M}$ (where by convention we have that $p_{L,M}$ corresponds to the $+1$ outcome, while $1 - p_{L,M}$ to the -1).
– If, for any pair (L, M) :

$$|p_{L,M} - \cos^2((L - M)\pi/8)| \geq \epsilon_2 \tag{G.2}$$

the client aborts.
Output: If the client does not abort (and this happens with non-negligible probability), then there exist an index-independent isometry Φ, given below in Fig. 5, such that

$$\Phi(|\Psi(L)\rangle) \overset{\epsilon(\epsilon_1,\epsilon_2)}{\approx} \left|+_{L\pi/4}\right\rangle \otimes |\Psi(000)\rangle \tag{G.3}$$

In Scenario 1, we make a number of assumptions. Firstly, the server actually knows the classical description of all eight possible states. This is done to simplify things (see scenario 2b), but has one disadvantage. The server can compute $\rho_L + \rho_{L\oplus 100}$ and unless $\Delta(\rho_L + \rho_{L\oplus 100}, \rho_{L'} + \rho_{L'\oplus 100}) \leq \epsilon_1$ where Δ is the trace distance, the server could guess the basis (with non-negligible advantage) by performing a minimum error measurement between these two mixed states. The measurement itself is likely to be of polynomial complexity, therefore it seems impossible to guarantee computational basis blindness in this setting (unless, of course, the states are also information theoretically close). For this reason, and to simplify the first exposition to *blind self-testing*, we give the proof by

[15] The distance represents either trace-distance in the information theoretic security setting or QPT distinguisher in the computational security setting.

assuming that Eq. G.1 is close in trace-distance in Section H. Now we proceed with the other scenarios of blind self-testing, so that the relevance of this notion for verifiable QFactory becomes apparent.

Protocol G.2 Blind self-testing: The independent non-identically distributed case (Scenario 2a)

– The server chooses N eight-plets of states $\{|\Psi(000)\rangle_i, |\Psi(001)\rangle_i, \cdots, |\Psi(111)\rangle_i\}_{i\in\{1,\cdots,N\}}$, such that they are basis blind, i.e.

$$\Delta(\rho_{i,L} + \rho_{i,L\oplus 100}, \rho_{i,L'} + \rho_{i,L'\oplus 100}) \leq \epsilon_1 \quad \forall\, i \in \{1, ..., N\}\ \forall\, L, L' \in \{0, ..., 7\} \tag{G.4}$$

where $\rho_{i,L} := |\Psi(L)\rangle_i \langle\Psi(L)|_i$.
– The client chooses randomly N indices, $\{L_1, \cdots, L_N\}$, where each $L_i \leftarrow \{000, \cdots, 111\}$ and sends the set of states $\{|\Psi(L_1)\rangle_1, |\Psi(L_2)\rangle_2, \cdots, |\Psi(L_N)\rangle_N\}$ to the server, while keeping the indices L_i secret.
– The client, randomly chooses a fraction f of the qubits to be used as tests and announces the set of corresponding indices $T = \{i_1, \cdots, i_{fN}\} \subset \{1, 2, ..., N\}$ to the server.
– For each test qubit $i_j \in T$, the client chooses a random measurement index $M_{i_j} \in \{000, \cdots, 111\}$ and instructs the server to measure the corresponding qubit in the $\left\{\left|+_{M_{i_j}\pi/4}\right\rangle, \left|-_{M_{i_j}\pi/4}\right\rangle\right\}$ basis.
– The server returns the test measurement results $\{c_{(i_j)}\}$.
– For each fixed pair (L, M), the client gathers all the test positions that correspond to that pair and from the relative frequencies, the client obtains an estimate for the probability $p_{L,M}$ (where by convention we have that $p_{L,M}$ corresponds to the $+1$ outcome, while $1 - p_{L,M}$ to the -1). Note, that each $p_{L,M}$ involves (in general) the statistics from different states.
– If $|p_{L,M} - \cos^2((L - M)\pi/8)| \geq \epsilon_2$ for any pair (L, M) the client aborts.
Output: If the client does not abort (and this happens with non-negligible probability), then there exists an index-independent isometry Φ, given below in Fig. 5, that if applied to a random non-tested qubit i, is acting in the following way:

$$\Phi(|\Psi(L)\rangle_i) \overset{\epsilon(\epsilon_1,\epsilon_2)}{\approx} \left|+_{L\pi/4}\right\rangle \otimes |\Psi(000)\rangle_i \tag{G.5}$$

We note that Scenario 2a is similar with scenario 1 with the crucial difference that different sets of eight states are used for each of the N qubits. It is not hard to see that very similar analysis with the one of Scenario 1 will apply, if instead of N different sets, one has N copies of the same state, but replaces that state with the average state defined to be $\rho_L(average) := \sum_{i=1}^{N} \rho_{i,L}$. The result will then hold with high probability using Hoeffding inequalities [Hoe63].

Protocol G.3 Blind self-testing: The independent non-identically distributed case (Scenario 2b)

– The server prepares N states $|\Psi(L)\rangle_i, i \in \{1,\cdots,N\}, L \in \{0,...,7\}$. For each of these states, the server is basis blind, i.e. (being computationally bounded) he cannot determine the two basis bits of the index L. On the other hand, the client does know the index L for each of the N states.[16]
– The client, randomly chooses a fraction f of the qubits to be used as tests and announces the set of corresponding indices $T = \{i_1,\cdots,i_{fN}\} \subset \{1,2,...,N\}$ to the server.
– For each test qubit $i_j \in T$, the client chooses a random measurement index $M_{i_j} \in \{000,\cdots,111\}$ and instructs the server to measure the corresponding qubit in the $\left\{\left|+_{M_{i_j}\pi/4}\right\rangle, \left|-_{M_{i_j}\pi/4}\right\rangle\right\}$ basis.
– The server returns the test measurement results $\{c_{(i_j)}\}$.
— For each fixed pair (L,M), the client gathers all the test positions that correspond to that pair and from the relative frequencies, the client obtains an estimate for the probability $p_{L,M}$ (where by convention we have that $p_{L,M}$ corresponds to the $+1$ outcome, while $1-p_{L,M}$ to the -1). Note, that each $p_{L,M}$ involves (in general) the statistics from different states.
– If $|p_{L,M} - \cos^2((L-M)\pi/8)| \geq \epsilon_2$ for any pair (L,M) the client aborts.
Output: If the client does not abort (and this happens with non-negligible probability), then there exist an index-independent polynomial isometry Φ, given below in Fig. 5, that if applied to a random non-tested qubit i is acting in the following way:

$$\Phi(|\Psi(L)\rangle_i) \overset{\epsilon(\epsilon_1,\epsilon_2)}{\approx} \left|+_{L\pi/4}\right\rangle \otimes |\Psi(000)\rangle_i \tag{G.6}$$

The crucial difference in scenario 2b, is that the server prepares only one state (not eight). In the previous scenarios, it was the client choosing which index L is used, and thus, unless the states $|\Psi(L)\rangle$ leaked information about the (basis bits of the) index, the server was blind. Here we impose this by requiring explicitly that the server prepares a state that he is basis-blind with respect to its index. There are two consequences of these differences.
First, now that the state is prepared on the server side, the client does not need to have any quantum ability, and his part in the protocol is purely classical. Second, since the state is prepared in the server's side, it is clear that we can no longer be in the information-theoretic setting, since, with unbounded computation power, he would be able to recover the exact label L. On the positive side, the issue we had in scenario 1, that the server could perform minimum error measurements is no longer valid, since the server does not know the classical description of the two states $(\rho_{i,L} + \rho_{i,L\oplus 100}; \rho_{i,L'} + \rho_{i,L'\oplus 100})$ that he needs to distinguish and thus cannot find the corresponding minimum error measurement. Finally, some care is needed to specify how the client can possibly know the index L while the server (that prepares the state) he does not. One way to achieve this is given

[16] This can be achieved either by having the client have exponential capacity, or by having the server prepare the states using some choice made by the client that has kept some side trapdoor information too.

in Malicious 8-states QFactory. Actually, scenario 2b corresponds exactly to the setting of Verifiable QFactory, provided that whatever deviation the server does in one round of Malicious 8-states QFactory is restricted (and independent) to other rounds.

Finally, we can generalise further (Protocol 8.1 removing the assumption of tensor produce states.

H Proof of Scenario 1: i.i.d. blind self-testing

Given the protocol G.1 (Scenario 1), we can assume that there exist eight untrusted, binary observables O'_M, with ± 1 eigenvalues, where we define $O'_0 := X', O'_{010} := Y'$. Each of these observable are of polynomial size, i.e. can be performed by a QPT party. The corresponding ideal observables are denoted without prime and we have $O_M = \left|+_{M\pi/4}\right\rangle\left\langle+_{M\pi/4}\right| - \left|-_{M\pi/4}\right\rangle\left\langle-_{M\pi/4}\right|$. We denote ρ_L the set of the eight (untrusted) states, and as defined in scenario 1 we consider the pure states (a purification in general) $|\Psi(L)\rangle$.

We can see that index independent isometry in Figure 5 gives us:

$$\Phi(|+\rangle|\Psi(L)\rangle) := |+\rangle\frac{(I+X')}{2}|\Psi(L)\rangle + |-\rangle(-iY')\frac{(I-X')}{2}|\Psi(L)\rangle \quad \text{(H.1)}$$

and we want to show that the state in Eq. (H.1) is $\overset{\epsilon}{\approx} \left|+_{L\pi/4}\right\rangle \otimes |\Psi(000)\rangle$ using the constrains coming from the basis blindness property and the measurement statistics of the test qubits. In this first exposition to the blind self-testing, we are going to prove the correctness up to ϵ, while the details of the robustness of this result (the explicit dependence of the final ϵ on ϵ_1, ϵ_2 appearing in the constraints) is left for future work.

Theorem 10. *If Protocol G.1 does not abort (with non-negligible probability) then the isometry given by Fig. 5 satisfies the condition of Eq. (G.3).*

To prove this theorem we first need two lemmas.

Lemma 13. *For any set of (eight) states $\{|\Psi(L)\rangle\}$ as from scenario 1, that is basis blind, i.e. $|(\rho_L+\rho_{L\oplus 100})-(\rho_{L'}+\rho_{L'\oplus 100})| \leq \epsilon_1$, all eight states belong (approximately) in a 2-dimensional subspace spanned by the vectors $\{|\Psi(000)\rangle, |\Psi(100)\rangle\}$.*

Proof. The binary observables O'_M have ± 1 eigenvalues, and can therefore be written as $O'_M = (+1)P^+_M + (-1)P^-_M$ where $P^\pm_M$ are the projections on the $+1$ and -1 eigenspace, respectively. It follows trivially that $(O'_M)^2 = I$ and that the corresponding projections are $P^+_M = \frac{I+O'_M}{2}$ and $P^-_M = \frac{I-O'_M}{2}$.

Moreover, if the protocol does not abort (and this happens with non-negligible probability), we also have the constraints on the expectation values of the observables coming from Eq. (G.2). From $\langle\Psi(L)|O'_L|\Psi(L)\rangle \overset{\epsilon_2}{\approx} 1$ and $\langle\Psi(L)|O'_{L\oplus 100}|\Psi(L)\rangle \overset{\epsilon_2}{\approx} -1$, we get that $O'_L|\Psi(L\oplus 100)\rangle \approx -|\Psi(L\oplus 100)\rangle$ and thus

$$P_L^+ |\Psi(L)\rangle \approx |\Psi(L)\rangle \ , \ P_L^- |\Psi(L)\rangle \approx 0$$
$$P_L^- |\Psi(L\oplus 100)\rangle \approx |\Psi(L\oplus 100)\rangle \ , \ P_L^+ |\Psi(L\oplus 100)\rangle \approx 0$$

which means that $\langle\Psi(L\oplus 100)|\Psi(L)\rangle \approx 0$. The space spanned by two vectors $\{|\Psi(L)\rangle, |\Psi(L\oplus 100)\rangle\}$ is two dimensional and for all L the state $1/2(\rho_L + \rho_{L\oplus 100})$ is the identity in that subspace. Now from the basis blindness condition we have that $1/2(\rho_L + \rho_{L\oplus 100}) \overset{\epsilon_1}{\approx} 1/2(\rho_{000} + \rho_{100})$, which means that all states $|\Psi(L)\rangle$ belong to that (fixed) 2-dimensional subspace. We will denote the projection on this 2-dimensional subspace as P_Ψ. □

Lemma 14. *The states given in Protocol G.1 are approximately of the form*

$$|\Psi(L)\rangle \overset{\epsilon}{\approx} \cos\left(\frac{L\pi}{8}\right)|\Psi(000)\rangle + e^{i\phi}\sin\left(\frac{L\pi}{8}\right)|\Psi(100)\rangle \tag{H.2}$$

with ϕ being a constant (independent of L). Furthermore, the untrusted operator Y' acts in the following way:

$$Y'|\Psi(100)\rangle = e^{-i\phi}|\Psi(000)\rangle \tag{H.3}$$

Proof. From Lemma 13 we know we can express all states in the following form:

$$|\Psi(L)\rangle \approx e^{if_2(L)}(a_0(L)|\Psi(000)\rangle + e^{if_1(L)}a_1(L)|\Psi(100)\rangle) \tag{H.4}$$

where $a_0(L), a_1(L), f_1(L)$ are functions to be determined and $f_2(L)$ is an overall complex phase that we could ignore, but we keep it here to remind that we can use this to simplify the final expressions. From the statistics of the measurements of the X' observable, we get (directly) that $a_0(L) \approx |\cos\frac{L\pi}{8}|$, $a_1(L) \approx |\sin\frac{L\pi}{8}|$.

$$|\Psi(L)\rangle \approx |\cos\frac{L\pi}{8}||\Psi(000)\rangle + e^{if_1(L)}|\sin\frac{L\pi}{8}||\Psi(100)\rangle \tag{H.5}$$

where we dropped the global phase $e^{if_2(L)}$. Now for $L \neq 000, 100$, from $\langle\Psi(L)|\Psi(L\oplus 100)\rangle \approx 0$, we obtain that:

$$f_1(L\oplus 100) = (f_1(L) + \pi) \mod 2\pi \tag{H.6}$$

and we can express the states grouped in four orthogonal bases:

$$\begin{aligned} &|\Psi(000)\rangle\,; \\ &|\Psi(001)\rangle = |\cos\frac{\pi}{8}||\Psi(000)\rangle + e^{if_1(001)}|\sin\frac{\pi}{8}||\Psi(100)\rangle\,; \end{aligned} \tag{H.7}$$

$$
\begin{aligned}
|\Psi(010)\rangle &= \frac{1}{\sqrt{2}}\left(|\Psi(000)\rangle + e^{if_1(010)}\,|\Psi(100)\rangle\right);\\
|\Psi(011)\rangle &= |\sin\frac{\pi}{8}|\,|\Psi(000)\rangle + e^{if_1(011)}|\cos\frac{\pi}{8}|\,|\Psi(100)\rangle\,;\\
|\Psi(100)\rangle\,;&\\
|\Psi(101)\rangle &= |\sin\frac{\pi}{8}|\,|\Psi(000)\rangle - e^{if_1(001)}|\cos\frac{\pi}{8}|\,|\Psi(100)\rangle\,;\\
|\Psi(110)\rangle &= \frac{1}{\sqrt{2}}\left(|\Psi(000)\rangle - e^{if_1(010)}\,|\Psi(100)\rangle\right);\\
|\Psi(111)\rangle &= |\cos\frac{\pi}{8}|\,|\Psi(000)\rangle - e^{if_1(011)}|\sin\frac{\pi}{8}|\,|\Psi(100)\rangle
\end{aligned}
\tag{H.8}
$$

where we used some identities such as $|\cos(3\pi/8)| = |\sin(\pi/8)|$, etc. Now, we have three parameters to fix, namely $f_1(001), f_1(010), f_1(011)$. For notational simplicity, we will use $c := |\cos\pi/8|\ ;\ s := |\sin\pi/8|$.

Then, we will use the statistics that we have from Eq. (G.2), when measuring in a different than the X' basis. Expressing $|\Psi(001)\rangle$ in the Y' basis we get:

$$
\begin{aligned}
|\Psi(001)\rangle = \frac{1}{\sqrt{2}}\left(c + se^{i(f_1(001)-f_1(010))}\right)|\Psi(010)\rangle +\\
\frac{1}{\sqrt{2}}\left(c - se^{i(f_1(001)-f_1(010))}\right)|\Psi(110)\rangle
\end{aligned}
\tag{H.9}
$$

From Eq. (G.2) and the probability of obtaining the result 010 when having the state $|\Psi(001)\rangle$ we obtain:

$$
\frac{1}{\sqrt{2}}|\left(c + se^{i(f(001)-f(010))}\right)| \approx c \tag{H.10}
$$

that is possible only if $e^{i(f(001)-f(010))} = 1$, i.e. we have:

$$
f_1(001) = f_1(010) \mod 2\pi \tag{H.11}
$$

Similarly, by expressing $|\Psi(011)\rangle$ in the Y' basis we get:

$$
\begin{aligned}
|\Psi(011)\rangle = \frac{1}{\sqrt{2}}\left(s + ce^{i(f_1(011)-f_1(010))}\right)|\Psi(010)\rangle +\\
\frac{1}{\sqrt{2}}\left(s - ce^{i(f_1(011)-f_1(010))}\right)|\Psi(110)\rangle
\end{aligned}
\tag{H.12}
$$

From Eq. (G.2) and the probability of obtaining the result 010 when having the state $|\Psi(011)\rangle$, we have:

$$\frac{1}{\sqrt{2}}|\left(s + ce^{i(f(011)-f(010))}\right)| \approx c \tag{H.13}$$

that is possible only if $e^{i(f(011)-f(010))} = 1$, i.e. we get:

$$f_1(011) = f_1(010) \mod 2\pi \tag{H.14}$$

Setting $f(001) := \phi$, we use Eqs. (H.6, H.11, H.14) and Eq. (H.7) becomes:

$$\begin{aligned}
&|\Psi(000)\rangle\,; \qquad (H.15)\\
&\quad |\Psi(001)\rangle = \cos\frac{\pi}{8}\,|\Psi(000)\rangle + e^{i\phi}\sin\frac{\pi}{8}\,|\Psi(100)\rangle\,;\\
&\quad |\Psi(010)\rangle = \cos\frac{2\pi}{8}\,|\Psi(000)\rangle + e^{i\phi}\sin\frac{2\pi}{8}\,|\Psi(100)\rangle\,;\\
&\quad |\Psi(011)\rangle = \cos\frac{3\pi}{8}\,|\Psi(000)\rangle + e^{i\phi}\sin\frac{3\pi}{8}|\,|\Psi(100)\rangle\,;\\
&|\Psi(100)\rangle\,;\\
&\quad |\Psi(101)\rangle = -\cos\frac{5\pi}{8}\,|\Psi(000)\rangle - e^{i\phi}\sin\frac{5\pi}{8}\,|\Psi(100)\rangle\,;\\
&\quad |\Psi(110)\rangle = -\cos\frac{6\pi}{8}\,|\Psi(000)\rangle - e^{i\phi}\sin\frac{6\pi}{8}\,|\Psi(100)\rangle\,;\\
&\quad |\Psi(111)\rangle = -\cos\frac{7\pi}{8}\,|\Psi(000)\rangle - e^{i\phi}\sin\frac{7\pi}{8}\,|\Psi(100)\rangle\,;
\end{aligned} \tag{H.16}$$

where we used $\cos\frac{5\pi}{8} = -\sin\frac{\pi}{8}$; $\sin\frac{5\pi}{8} = \cos\frac{\pi}{8}$; $-\cos\frac{6\pi}{8} = \frac{1}{\sqrt{2}}$; $\sin\frac{6\pi}{8} = \frac{1}{\sqrt{2}}$; $\cos\frac{3\pi}{8} = \sin\frac{\pi}{8}$; $\sin\frac{3\pi}{8} = \cos\frac{\pi}{8}$; $-\cos\frac{7\pi}{8} = \cos\frac{\pi}{8}$ and $\sin\frac{7\pi}{8} = \sin\frac{\pi}{8}$. By noting that each state is invariant if multiplied by a global phase (an overall minus sign in our case) we get the expression:

$$|\Psi(L)\rangle = \cos\left(\frac{L\pi}{8}\right)|\Psi(000)\rangle + e^{i\phi}\sin\left(\frac{L\pi}{8}\right)|\Psi(100)\rangle \tag{H.17}$$

as required for Lemma 14. The action of the observable Y' projected on the subspace that all $|\Psi(L)\rangle$ belong to, is:

$$P_\Psi Y' P_\Psi = |\Psi(010)\rangle\langle\Psi(010)| - |\Psi(110)\rangle\langle\Psi(110)| \tag{H.18}$$

and given that

$$|\Psi(010)\rangle = \frac{1}{\sqrt{2}}\left(|\Psi(000)\rangle + e^{i\phi}\,|\Psi(100)\rangle\right)$$

$$|\Psi(110)\rangle = \frac{1}{\sqrt{2}}\left(-|\Psi(000)\rangle + e^{i\phi}|\Psi(100)\rangle\right) \tag{H.19}$$

we get:

$$\begin{aligned} P_\Psi Y' P_\Psi &= e^{-i\phi}|\Psi(000)\rangle\langle\Psi(100)| + e^{i\phi}|\Psi(100)\rangle\langle\Psi(000)| \\ Y'|\Psi(100)\rangle &= e^{-i\phi}|\Psi(000)\rangle \end{aligned} \tag{H.20}$$

p □

Proof of Theorem 10. We can now prove Theorem 10. If the protocol does not abort, it means that the condition on the test measurement statistics is satisfied, and the above Lemmas hold. We substitute Eq. (H.2) in the isometry, note that $\frac{(I+X')}{2}|\Psi(000)\rangle = 1 = \frac{(I-X')}{2}|\Psi(100)\rangle$ and $\frac{(I+X')}{2}|\Psi(100)\rangle = 0 = \frac{(I-X')}{2}|\Psi(000)\rangle$ and use Eq. (H.3) to get the desired result:

$$\begin{aligned} \Phi(|+\rangle|\Psi(L)\rangle) &= |+\rangle\frac{(I+X')}{2}|\Psi(L)\rangle + |-\rangle(-iY')\frac{(I-X')}{2}|\Psi(L)\rangle \\ &= |+\rangle\cos\left(\frac{L\pi}{8}\right)|\Psi(000)\rangle + |-\rangle(-iY')e^{i\phi}\sin\left(\frac{L\pi}{8}\right)|\Psi(100)\rangle \\ &= |+\rangle\cos\left(\frac{L\pi}{8}\right)|\Psi(000)\rangle + |-\rangle e^{i\phi}\sin\left(\frac{L\pi}{8}\right)(-ie^{-i\phi})|\Psi(000)\rangle \\ &= \left(\cos\left(\frac{L\pi}{8}\right)|+\rangle - i\sin\left(\frac{L\pi}{8}\right)|-\rangle\right)|\Psi(000)\rangle \\ &= \left|+_{L\pi/4}\right\rangle|\Psi(000)\rangle \end{aligned} \tag{H.21}$$

□

I Generalisation to pseudo-homomorphic functions

I.1 Notation and remarks about the generalisation

Note that later on, we will do a slight abuse of notation, and if $\mathcal{K}$ is a set of keys, we will denote by $k \leftarrow \mathcal{K}$ the sampling of a key in $\mathcal{K}$. Note that this sampling may not be uniform, and depends on a fixed distribution.

Moreover, in order to have a function usable in practice, we would like to be able to create the uniform superposition on all the elements of the input set. However, in practice, it may not be possible to have an exact superposition on all the elements of this input set. Therefore we will consider in the following that the function g has an input set $\mathcal{Z}$ that is a bit bigger, but so that we can create an exact uniform superposition on all the elements of this set, and in order to deal with the fact that $\mathcal{Z}$ is not the initial input set, we will say that $g(z) = \perp$ as soon as z does not belong to the initial input set of g. Note that we may also

abuse notation for simplicity, and also write $g(z) = \perp$ when z does not even belong to $\mathcal{Z}$ and when g is not defined on a given input z.

We will also need to extend some notions to this new notion. For example, we will say that a function $g : \mathcal{Z} \to \mathcal{Y} \cup \perp$ is **injective** when for all $y \in \mathcal{Y}$, $|f^{-1}(y)| \leq 1$.

I.2 Definition

Definition 14 ($(\eta, \mathcal{Z}, \mathcal{Z}_0, \mathcal{D})$-homomorphic family of functions). *Let us consider a family of functions $\{g_k \colon \mathcal{Z} \to \mathcal{Y} \cup \perp\}_{k \in \mathcal{K}}$, as well as two symmetric binary group relations $*$ and $\star$, with $*$ acting on a set containing $\mathcal{Z}$ and $\mathcal{Z}_0$, $\star$ acting on $\mathcal{Y} \cup \perp$, and so that $\forall y \in \mathcal{Y}, \perp \star y = \perp$. We say that $\{f_k\}_{k \in \mathcal{K}}$ is an (η, $\mathcal{Z}$, $\mathcal{Z}_0$, $\mathcal{D}$)-homomorphic function if $\mathcal{D}$ is a distribution on $\mathcal{Z}_0$ and*

$$\Pr_{\substack{k \leftarrow \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [z * z_0 \in \mathcal{Z} \text{ and } g_k(z) \star g_k(z_0) = g_k(z * z_0) \neq \perp] \geq \eta$$

Note that we do require that z is sampled uniformly from $\mathcal{Z}$, but z_0 is sampled from a distribution $\mathcal{D}$ on $\mathcal{Z}_0$ that may not be uniform.

Definition 15 (δ-2-regular family of functions). *Let us consider a family of functions $\{f_k \colon \mathcal{X} \to \mathcal{Y} \cup \perp\}_{k \in \mathcal{K}}$. For a fixed k, $\mathcal{Y}^{(2)}$ will be the set of y having two preimages: $\mathcal{Y}^{(2)}_{f_k} = \{y \in \mathcal{Y}, |f_k^{-1}(y)| = 2\}$. Then, this family of functions is said to be δ-2-regular if*

$$\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ x \leftarrow\$ \mathcal{X}}} [f_k(x) \in \mathcal{Y}^{(2)}_{f_k}] \geq \delta$$

Lemma 15 ($(\eta, \mathcal{Z}, \mathcal{Z}_0)$-homomorphy to δ-2-regularity). *Given a family of functions $\{g_k \colon \mathcal{Z} \to \mathcal{Y} \cup \perp\}_{k \in \mathcal{K}}$ that is both injective and an (η, $\mathcal{Z}$, $\mathcal{Z}_0$)-homomorphic family of functions, then it's possible to build a family $\{f_{k'} \colon \mathcal{Z} \times \{0,1\} \to \mathcal{Y} \cup \perp\}_{k' \in \mathcal{K}'}$ that is δ-2-regular, with $\delta = \eta$.*

Proof. Let's do the following construction. To sample a key $k' \in \mathcal{K}'$, we first sample a key k from $\mathcal{K}$, as well as an $z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0$, and we define $k' = (k, y_0 := f_k(z_0))$. Then, we define $f_{k'}(z, 0) = g_k(z)$ and $f_{k'}(z, 1) = g_k(z) \star y_0$, also denoted later as $f_{k'}(z, c) = g_k(z) \star (c \cdot y_0)$ for simplicity. Now, we remark that:

$$\Pr_{\substack{k' \leftarrow\$ \mathcal{K}' \\ x \leftarrow\$ \mathcal{X}}} [f_{k'}(x) \in \mathcal{Y}^{(2)}_{f'_k}] \tag{I.1}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z} \\ c \leftarrow\$ \{0,1\}}} [g_k(z) \star (c \cdot g_k(z_0)) \in \mathcal{Y}^{(2)}_{f'_k}] \tag{I.2}$$

$$= \frac{1}{2} \times \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \in \mathcal{Y}^{(2)}_{f'_k}] + \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \star g_k(z_0) \in \mathcal{Y}^{(2)}_{f'_k}] \Big) \tag{I.3}$$

$$\geq \frac{1}{2} \times \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \in \mathcal{Y}_{f'_k}^{(2)} \text{ and } \underbrace{z * z_0^{-1} \in \mathcal{Z} \text{ and } g_k(z) = g_k(z * z_0^{-1}) \star g_k(z_0) \neq \perp}_{C_1}] \tag{I.4}$$

$$+ \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)} \text{ and } \underbrace{z * z_0 \in \mathcal{Z} \text{ and } g_k(z) \star g_k(z_0) = g_k(z * z_0) \neq \perp}_{C_2}] \Big) \tag{I.5}$$

$$= \frac{1}{2} \times \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \in \mathcal{Y}_{f'_k}^{(2)} | C_1] \times \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_1] \tag{I.6}$$

$$+ \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)} | C_2] \times \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_2] \Big) \tag{I.7}$$

Now, remark that when $z_0 * z \in \mathcal{D}$ and $g_k(z_0) \star g_k(z) = g_k(z_0 * z) \neq \perp$, then $y := g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)}$. Indeed:

- $y \in \mathcal{Y}$ because $g_k(z) \star g_k(z_0) \neq \perp$ and the $\star$ operator is defined on $\mathcal{Y} \cup \perp$
- there are at least two preimages mapping to y, because $y = f_k(z, 1) = g_k(z) \star g_k(z_0) = g_k(z * z_0) = f_k(z * z_0, 0)$.
- there are at most two preimages mapping to y: indeed g_k is injective, so both partial functions $f(\cdot, 0)$ and $f(\cdot, 1)$ are injective, so it's not possible to have more than two preimages mapping to y.

So $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \star g_k(z_0) \in \mathcal{Y}_{f'_k}^{(2)} | C_2] = 1$. Similarly, $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [g_k(z) \in \mathcal{Y}_{f'_k}^{(2)} | C_1] =$ 1. So we can rewrite the above equation as:

$$\Pr_{\substack{k' \leftarrow\$ \mathcal{K}' \\ x \leftarrow\$ \mathcal{X}}} [f_{k'}(x) \in \mathcal{Y}_{f'_k}^{(2)}] \tag{I.8}$$

$$\geq \frac{1}{2} \times \Big(\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_1] + \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_2] \Big) \tag{I.9}$$

Now, remember that $\{g_k\}_k$ is $(\eta,\ \mathcal{Z},\ \mathcal{Z}_0)$-homomorphic, so $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_2] \geq \eta$.

By symmetry, we also have $\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_1] \geq \eta$. Indeed:

$$\Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [\underbrace{z * z_0^{-1}}_{\hat{z}} \in \mathcal{Z} \text{ and } g_k(z) = g_k(z * z_0^{-1}) \star g_k(z_0) \neq \perp] \tag{I.10}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [\hat{z} \in \mathcal{Z} \text{ and } z \in \mathcal{Z} \text{ and } \hat{z} = z * z_0^{-1} \text{ and } g_k(z) = g_k(\hat{z}) \star g_k(z_0) \neq \perp] \tag{I.11}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ \hat{z} \leftarrow\$ \mathcal{Z}}} [\hat{z} \in \mathcal{Z} \text{ and } z \in \mathcal{Z} \text{ and } \hat{z} = z * z_0^{-1} \text{ and } g_k(z) = g_k(\hat{z}) \star g_k(z_0) \neq \perp] \tag{I.12}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ \hat{z} \leftarrow\$ \mathcal{Z}}} [\hat{z} * z_0 \in \mathcal{Z} \text{ and } g_k(\hat{z} * z_0) = g_k(\hat{z}) \star g_k(z_0) \neq \perp] \tag{I.13}$$

$$= \Pr_{\substack{k \leftarrow\$ \mathcal{K} \\ z_0 \leftarrow^{\mathcal{D}} \mathcal{Z}_0 \\ z \leftarrow\$ \mathcal{Z}}} [C_2] \tag{I.14}$$

$$\geq \eta \tag{I.15}$$

So $\Pr_{\substack{k' \leftarrow\$ \mathcal{K}' \\ x \leftarrow\$ \mathcal{X}}}[f_{k'}(x) \in \mathcal{Y}_{f'_k}^{(2)}] \geq \eta$, which concludes the proof. □

J Proof of the Malicious-Abort QFactory

In this section we moved some of the proofs of the security and correctness of Protocol 7.1, which are summarised in Theorem 7.

Proof of Lemma 4. The function f_k cannot have more than two preimages by assumption, and in the Malicious 4-states QFactory protocol the output y is in the image of f_k. So it means that y has exactly one preimage x. So after measuring the last register, the states will be in the state $|0\rangle \otimes |x\rangle \otimes |y\rangle$. Then, we apply U_h, so the states becomes $|d\rangle \otimes |x\rangle \otimes |y\rangle$ with $d \in \{0, 1\}$. We remark that the first qubit is not entangled with the measured qubits, so the output qubit will be $|d\rangle$, which is indeed in the basis $\{|0\rangle, |1\rangle\}$. □

Proof of Lemma 5. The entire analysis of the circuit will be performed only with respect to the basis of the states of the circuit. Let us first examine the first part of the circuit, where we apply $\wedge Z$ between $\left|+_{\frac{\pi}{2}}\right\rangle$ and $|\mathtt{in}_1\rangle = H^{{B_1}^{(1)}} Z^{{B_2}^{(1)}}$ (with ${B_1}^{(1)}$ the basis of $|\mathtt{in}_1\rangle$) and then measure the first qubit in the $|\pm\rangle$ basis, and we denote the result state V_1.
The result of this operation is:
- if ${B_1}^{(1)} = 0$, $V_1 = R(\pi(B_2^{(1)} + s_{1,1} + 1)) \left|+_{\frac{\pi}{2}}\right\rangle \in \{\left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$
- if ${B_1}^{(1)} = 1$, $V_1 = X^{B_2^{(1)}} |0\rangle \in \{|0\rangle, |1\rangle\}$
In other words, the state V_1 belongs to the basis $\mathcal{B}_0 = \{\left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$, if ${B_1}^{(1)} = 0$ and to the basis $\mathcal{B}_1 = \{|0\rangle, |1\rangle\}$ if ${B_1}^{(1)} = 1$.
Now, we can think of the circuit as having t states $V_i \in \{|0\rangle, |1\rangle, \left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$, where every V_i has the basis ${B_1}^{(i)}$. Then, to compute the output state $|\mathtt{out}\rangle$ of $\mathtt{Gad}_{\oplus}$, for every $i \in \{1, ..., t\}$ we have to apply CZ between V_i and $|+\rangle$ and then measure the first qubit in the $|\pm\rangle$ basis.
So let us do this step first for V_1. The result is a state $W_1 = X^{s_{1,2}} H V_1$, thus we obtain that:
W_1 belongs to the basis $\mathcal{B}_0 = \{\left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$, if ${B_1}^{(1)} = 0$ and to the basis

$\mathcal{B}_2 = \{|+\rangle, |-\rangle\}$ if ${B_1}^{(1)} = 1$.
Next we do the same operations between V_2 and W_1, the result being a state W_2, then between V_3 and W_2 and so on, therefore, the outcome state is $|\mathtt{out}\rangle = W_t$. We will prove by induction that the state $W_t \in \{|+\rangle, |-\rangle, \left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$, where the basis of W_t is given by $B_1 = {B_1}^{(1)} \oplus ... \oplus {B_1}^{(t)}$.
As we have proved already for the basis case $t = 1$, we now prove the induction step. Suppose that $W_n \in \{|+\rangle, |-\rangle, \left|+_{\frac{\pi}{2}}\right\rangle, \left|-_{\frac{\pi}{2}}\right\rangle\}$ with basis $B_1 = {B_1}^{(1)} \oplus ... \oplus {B_1}^{(n)}$.
To obtain W_{n+1} we have to apply $\wedge Z$ between V_{n+1} and W_n and then measure the first qubit. Then after computing this, we obtain that the basis of W_{n+1} is B_1 if the basis of V_{n+1} is ${B_1}^{(n+1)} = 0$ and the basis of W_{n+1} is $1 \oplus B_1$ if the basis of V_{n+1} is ${B_1}^{(n+1)} = 1$. In other words, the basis of W_{n+1} is given by $B_1 = {B_1}^{(1)} \oplus ... \oplus {B_1}^{(n)} \oplus {B_1}^{(n+1)}$, which concludes the proof. □

Proof of Lemma 6. In the honest case, all runs are independents, so let us define $\{A_i\}_{i=1}^{t}$ as the (binary) random variables whose values are 1 iff the i-th run has two preimages associated with y_i. We know that for all i, $\mathbb{E}(A_i) \geq p_a > p_b$. So let us define $\varepsilon = \mathbb{E}(A_i) - p_b > p_a - p_b$. Using Chernoff inequality we have

$$\Pr\left[\frac{1}{t}\sum_{i=1}^{t} A_i < \mathbb{E}(A_i) - \varepsilon\right] \leq e^{-2\varepsilon^2 t} \leq e^{-2(p_a - p_b)^2 t} = \mathsf{negl}(t)$$

(because $p_a - p_b$ is constant) □

Proof of Lemma 7. The Lemma 6 gives that the probability to have more than $p_c t_c$ accepted runs for a given chunk is $1 - \mathsf{negl}(t_c)$, i.e. if $t_c = \Omega(n)$, this probability is $\mathsf{negl}(n)$. So for n_c chunks, the probability to have one fail is $(1 - \mathsf{negl}(n))^{n_c} = 1 - \mathsf{negl}(n)$ as soon as $n_c = \mathsf{poly}(n)$, which is the case because $t = t_c \times n_c = \mathsf{poly}(n)$. Then, when all the chunks are accepted, the correctness of the output values is assured by Lemma 5. □

Proof of Lemma 8. By contradiction, let us assume that there is an adversary $\mathcal{A}$ such that (we omit the parameters for readability)

$$\Pr\left[\tilde{B}_1 = \beta\right] > \eta$$

Then, if we define $a_i := a(k^{(i)}, y^{(i)})$,

$$\begin{aligned}\eta &< \Pr\left[\tilde{B}_1 = \beta\right] \\ &= \underbrace{\Pr\left[\sum_i a_i < p_c t_c\right]}_{\alpha} \times \frac{1}{2} + \Pr\left[\sum_i a_i \geq p_c t_c\right] \times \Pr\left[\tilde{B}_1 = \beta \mid \sum_i a_i \geq p_c t_c\right]\end{aligned}$$

$$= \alpha \times \frac{1}{2} + (1-\alpha) \times \Pr\left[\tilde{B}_1 = \beta \mid \sum_i a_i \geq p_c t_c\right]$$
$$\leq \alpha \times \frac{1}{2} + (1-\alpha) = 1 - \frac{\alpha}{2}$$

so $\alpha \leq 2(1-\eta)$.

Now, we remark that we can bound also $(1-a) \times \Pr\left[\tilde{B}_1 = \beta \mid \sum_i a_i \geq p_c t_c\right]$. Indeed, if this value is too big then we can construct an adversary that could break the hardcore bit property of g_K. To do that, we define an adversary $\mathcal{A}'$ taking as input a k, and whose goal is to define the hardcore bit d_0 associated with k. This adversary will pick $t_c - 1$ public keys/trapdoors $(k^{(i)}, t_{k^{(i)}})$, and hide k in the middle of these trapdoors. Then, $\mathcal{A}'$ calls $\mathcal{A}$ with these t_c keys, and outputs $\tilde{d_0} := \tilde{B}_1 \oplus_i a^{(i)} d_0^{(i)}$, with $\tilde{B}_1$ the output of $\mathcal{A}$, and $a^{(i)}$ computed by using the $y^{(i)}$ provided by $\mathcal{A}$. We know that $\tilde{d_0} = d_0$ when the guess of $\mathcal{A}'$ was right, when $\sum_i a_i \geq p_c t_c$, and when the y corresponding to the function k has two preimages. But this even occurs with probability greater than $(1-\alpha) \times \Pr\left[\tilde{B}_1 = \beta \mid \sum_i a_i \geq p_c t_c\right] \times p_c$, and because d_0 is a hardcore bit, this probability is bounded by $1/2 + \mathsf{negl}(n)$, or equivalently:

$$(1-\alpha) \times \Pr\left[\tilde{B}_1 = \beta \mid \sum_i a_i \geq p_c t_c\right] \leq \frac{1}{2p_c} + \mathsf{negl}(n)$$

Now, let's come back to our probability to guess β:

$$\begin{aligned}\Pr\left[\tilde{B}_1 = \beta\right] &= \alpha \times \frac{1}{2} + (1-\alpha) \times \Pr\left[\tilde{B}_1 = \beta \mid \sum_i a_i \geq p_c t_c\right] \\ &\leq \alpha \times \frac{1}{2} + \frac{1}{2p_c} + \mathsf{negl}(n) \\ &\leq 1 - \eta + \frac{1}{2p_c} + \mathsf{negl}(n)\end{aligned}$$

But on the other side, $\Pr\left[\tilde{B}_1 = \beta\right] > \eta$, so

$$\begin{aligned}\eta &< 1 - \eta + \frac{1}{2p_c} + \mathsf{negl}(n) \\ \eta &< \frac{1}{2}\left(1 + \frac{1}{2p_c}\right) + \mathsf{negl}(n)\end{aligned}$$

Because η and p_c are constants[17] that do not depend on n, this equality is also true without the $\mathsf{negl}(n)$:

$$\eta < \frac{1}{2}\left(1 + \frac{1}{2p_c}\right)$$

which is absurd because $\eta = \frac{1}{2}\left(1 + \frac{1}{2p_c}\right)$. □

Proof of Theorem 7. The proof of correctness is made Lemma 7, and the security is a direct application of Conjecture 1: after using Lemma 8: this theorem provides a η such that it's not possible to solve one chunk with probability better than $\eta < 1$, so $\delta(n) := 1 - \eta$ is a constant (and $\delta(n) \geq \frac{1}{\mathsf{poly}(n)}$). Therefore Conjecture 1 tells us that no adversary can get the XOR of n_c chunks with probability better than $\frac{1}{2} + \eta^{n_c} + \mathsf{negl}(n)$. But $t_c = \Omega(n)$ and η is a constant, so no adversary can get the XOR of n_c chunks with probability better than $\frac{1}{2} + \mathsf{negl}(n)$, i.e. no adversary can find B_1 with probability better then $\frac{1}{2} + \mathsf{negl}(n)$. □

K Discussion about dealing with the abort case without Yao's XOR Lemma

The proof of security we have when we cannot assume that abort arrives with negligible probability defined in Section 7 has two drawbacks: first it relies on the conjecture that Yao's XOR Lemma is valid for one-round protocols (classical messages) with quantum adversary, but it also complicate the protocols by adding replication. We are also working on a second method that runs this time just one instance of Malicious 4-states QFactory, and will leaks the abort bit to the server. The specificity of this method is that the hash function will be send after receiving the y, and will be a 2-universal hash function (we just chose the function corresponding to the XOR of a random subset of hardcore bits that is easy to implement on server side as well). We also require that the δ-2-regular trapdoor family of functions needs to have a polynomial number of homomorphic hardcore bits (we can easily extend our courstruction to have such requirements by just adding $q/2 \cdot (d_{0,1}, \ldots, d_{0,t}, 0, \ldots, 0)^T$). This setting will bring us back very close to the requirements needed by the leftover hash lemma [IALL89]. The advantage of this method is that we have a quantum equivalent of this lemma [TSSR10], but unfortunately this lemma is valid (and expressed) in the information theoretic framework. Because our familly will never be information theoretically secure, but only computationally secure, we need an intermediate step to turn our information theoretically secure argument into a computationally secure one. Usually, this is done by introducing lossy functions [PW07], and then using some indistinguishable property between injective and lossy functions

[17] note that if we give them a dependence on n, we can make sure that $\eta - \frac{1}{2}\left(1 + \frac{1}{2p_c}\right)$ is non negligible, but for simplicity we will keep them constant

to finish the proof. Unfortunately, our protocol has a non standard shape, and it's not yet clear how to define a in the lossy case to make sure an adversary cannot exploit a to distinguish between lossy and non lossy.

We also point out that we can also create some constructions where the abort bit is independent of the secret, by first sending a function whose goal is to create the superposition, and after receiving the y, if y has two preimages, we send a single element with no noise $As_0 + q/2 \cdot d_0$ that will be used like if it were another raw in the y_0. This problem is still supposed to be difficult, and because the second message does not have noise, it cannot lead to an abort, to the abort is independent of the secret. But the proof of security for this construction is also a work in progress.

Finally, a way to boost the probability of success from polynomial to exponential without leaking the abort bit could be to consider "abort" cases as just errors in the transmission channels, and then to use error correcting code to correct them, but again, this approach is a work in progress.