QFactory: classically-instructed remote secret qubits preparation

Citation for published version:
Cojocaru, A, Colisson, L, Kashefi, E & Wallden, P 2019, QFactory: classically-instructed remote secret qubits preparation. in SD Galbraith & S Moriai (eds), Advances in Cryptology – ASIACRYPT 2019. Lecture Notes in Computer Science (LNCS), Springer, Cham, pp. 615-645, 25th Annual International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, 8/12/19. https://doi.org/10.1007/978-3-030-34578-5_22

Digital Object Identifier (DOI):
10.1007/978-3-030-34578-5_22

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Advances in Cryptology – ASIACRYPT 2019

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

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 [46] (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 [5, 14, 2, 10, 20, 41, 34, 24, 40, 22].

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 [11, 18, 3], or by defining their direct quantum analogues [30, 44, 50, 29]. Different encodings, on the client side, could also reduce the quantum communication [34, 24]. 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 [4, 54, 1, 42], 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 [32] 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 [7], 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 [8], 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 [21], and the above ideas to generate a single qubit "blind measurement device", [33] 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 [15] 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 [12] or Abstract Cryptography (AC) [35] should be possible (but left as a future work).

Concurrently with our work, [23] also took this modular approach. Technically they followed closely the ideas from [32, 33, 8], but the basic primitive they derive (in a verifiable version) is the one we introduced in [15]. 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 [23] 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 family 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 ([7]).

Following the modularity of [15], 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 [53].

1.1 Our Contributions

In [15] 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 computationally secure (against *quantum* adversaries). However, the proof of security was done in a weak model called "honest-but-curious". In this paper (full version at [16]):

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 [32], the second category of functions is strictly more secure and more standard in the cryptographic literature [7]. 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 [33], 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:

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

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 [15] we gave specific choices that achieve these properties. The starting point is the injective, trapdoor one-way family of functions $\bar{\mathcal{G}}$ from [39], 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 [39] (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

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

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

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

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:

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

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 [10], 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 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 [21] and directly relates our work with that of [33]. 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.

[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).

2 Preliminaries

We assume basic familiarity with quantum notions, a good reference is [43].
For a state $\left|+_\theta\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{i\theta}\left|1\right\rangle)$, 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 $\left|+_\theta\right\rangle$ 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 $\{\left|0\right\rangle, \left|1\right\rangle, \left|+\right\rangle, \left|-\right\rangle\}$, 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 $\left|0\right\rangle$ or $\left|1\right\rangle$, and $B_2 = 0$ if and only if the state is $\left|0\right\rangle$ or $\left|+\right\rangle$, i.e. $H^{B_1}X^{B_2}\left|0\right\rangle$. 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 $\{f_k : \mathcal{D} \rightarrow \mathcal{R}\}_{k \in \mathcal{K}}$ 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 $\{f_k : \mathcal{D} \rightarrow \mathcal{R}\}_{k \in \mathcal{K}}$ 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:*
  $\Pr_{\substack{k \leftarrow Gen(1^n) \\ rc \leftarrow \{0,1\}^*}} [\mathcal{A}(k) = (x, x') \text{such that } x \neq x' \text{ and } f_k(x) = f_k(x')] \leq \mathsf{negl}(n)$
  where rc is the randomness of $\mathcal{A}$ (rc will be omitted from now).

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

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

- *There exists a PPT algorithm* `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* `Pub`*, i.e. $k = \mathtt{Pub}(t_k)$*
- *$\{f_k : \mathcal{D} \rightarrow \mathcal{R}\}_{k \in \mathcal{K}}$ is a family of one-way functions;*
- *There exists a PPT algorithm* `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 $hc \colon \mathcal{D} \rightarrow \{0, 1\}$ 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 $\mathbb{Z}_q$.*

[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.

Regev [47] and Peikert [45] 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, [47, 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:*

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

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

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* 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:*

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

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 the full paper [16] 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 $(k, t_k) \leftarrow \text{Gen}_{\mathcal{F}}(1^n)$.
– Client: instructs Server to prepare one register at $\otimes^n H \left|0\right\rangle$ and second register initiated at $\left|0\right\rangle^m$.
– 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 $(\left|x\right\rangle + \left|x'\right\rangle) \otimes \left|y\right\rangle$ with $f_k(x) = f_k(x') = y$ and $y \in \operatorname{Im} f_k$.

Stage 2: Output preparation
– Server: applies U_{h_k} on the preimage register $\left|x\right\rangle + \left|x'\right\rangle$ as control and another qubit initiated at $\left|0\right\rangle$ as target. Then, measures all the qubits, but the target in the $\{\frac{1}{\sqrt{2}}(\left|0\right\rangle \pm \left|1\right\rangle)\}$ 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 $\left|\texttt{out}\right\rangle = H^{B_1} X^{B_2} \left|0\right\rangle$, 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 $\left|\texttt{out}\right\rangle$, 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 $y \in \operatorname{Im} f_k$ 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 $\{\left|0\right\rangle, \left|1\right\rangle, \left|+\right\rangle, \left|-\right\rangle\}$ and its exact description is the following:

Theorem 2. *In an honest run, with overwhelming probability the output state* $\left|\texttt{out}\right\rangle$ *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* $\left|h_k(x)\right\rangle$ *(computational basis, also equal to* $\left|h_k(x')\right\rangle$*)*
- *if* $d_0 = 1$*, then if* $\sum_i b_i \cdot (x_i \oplus x'_i) = 0 \bmod 2$*, the state is* $\left|+\right\rangle$*, otherwise the state is* $\left|-\right\rangle$ *(Hadamard basis).*

i.e.

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

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:

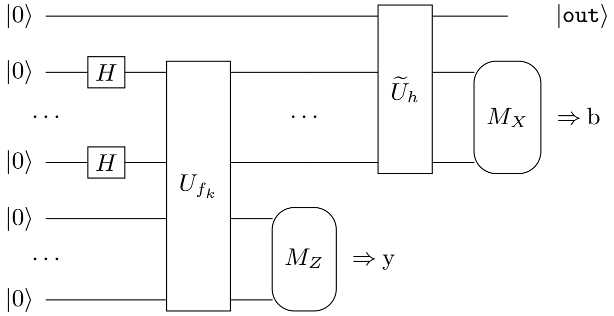

Fig. 1. The circuit computed by the Server

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

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

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

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

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

□

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 ([10, 22]) basis blindness is sufficient as indicated in [19]), 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 $1/2 + \mathsf{negl}(n)$. $\square$

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 the full paper), 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 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 $\mathcal{G}$ and $\mathcal{D}$ are homomorphic, there exist 2 operations " $+_{\mathcal{D}}$ " acting on $\mathcal{D}$ and " $+_{\mathcal{R}}$ " acting on $\mathcal{R}$ such that $\forall k, \forall z_1, z_2 \in \mathcal{D}$, $g_k(z_1 +_{\mathcal{D}} z_2) = g_k(z_1) +_{\mathcal{R}} g_k(z_2)$ and $h(z_1) \oplus h(z_2) = h(z_2 +_{\mathcal{D}} z_1) = h(z_2 -_{\mathcal{D}} z_1)$.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 (see $\mathtt{Gen}_{\mathcal{F}}(1^n)$ on the right). And then

```
$\mathtt{Gen}_{\mathcal{F}}(1^n)$
$(k, t_k) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$
$z_0 \leftarrow_{\$} \mathcal{D}$
$y_0 = g_k(z_0)$
$t'_{k'} = (t_k, z_0)$
$k' = (k, y_0)$
**return** $(k', t'_{k'})$
```

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, it is easy to see that this family is 2-regular homomorphic-hardcore family, as proven in the full paper.

□

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 [15] a similar construction, but without the additional homomorphic-hardcore property.

The starting point is the injective, trapdoor one-way family of functions $\bar{\mathcal{G}} = \{\bar{g}_K : \mathbb{Z}_q^n \times E^m \to \mathbb{Z}_q^m\}_K$[7] from [39] (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 [39], 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

$$y_0 := Ks_0 + e_0 + d_0 \times v$$

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 $\mathcal{A}$,*

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

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:

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

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

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:

- 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 [15]). *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:*

$$\begin{aligned} q &= 2^{5\lceil log(n)\rceil + 21} \\ m &= 23n + 5n\lceil log(n)\rceil \\ \mu &= 2mn\sqrt{23 + 5log(n)} \\ \alpha' &= \frac{\mu}{m\sqrt{mq}} \end{aligned} \tag{4.1}$$

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 the full paper, 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 functionality for applications such as blind quantum computing [10], we need to be able to produce states from the set $\{\left|+_\theta\right\rangle, \theta \in \{0, \frac{\pi}{4}, ..., \frac{7\pi}{4}\}\}$, 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.[8]

Protocol 5.1 Malicious 8-states QFactory

Requirements: Same as in Protocol Protocol 3.1
Input: Client runs twice 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 $\left|\mathtt{in}_1\right\rangle$ and a ”rotated” Malicious 4-states QFactory to obtain a state $\left|\mathtt{in}_2\right\rangle$ (by rotated Malicious 4-states QFactory we mean a Malicious 4-states QFactory, but where the last set of measurements in the $\left|\pm\right\rangle$ basis (Fig. 1) is replaced by 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 $\left|\mathtt{in}_1\right\rangle$ and (B'_1, B'_2) for $\left|\mathtt{in}_2\right\rangle$.
– Client: instructs Server to apply the Merge Gadget (Fig. 2) on the states $\left|\mathtt{in}_1\right\rangle$, $\left|\mathtt{in}_2\right\rangle$.
– 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:

$$\left|\mathtt{out}\right\rangle = X^{(s_2+B_2)\cdot B_1} Z^{B'_2+B_2(1-B_1)+B_1[s_1+(s_2+B_2)B'_1]} R(\frac{\pi}{2})^{B_1} R(\frac{\pi}{4})^{B'_1} \left|+\right\rangle \tag{5.1}$$

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 $\left|\mathtt{in}_1\right\rangle$ and $\left|\mathtt{in}_2\right\rangle$, where $\left|\mathtt{in}_1\right\rangle \in \{\left|0\right\rangle, \left|1\right\rangle, \left|+\right\rangle, \left|-\right\rangle\}$ and $\left|\mathtt{in}_2\right\rangle \in \{\left|+\right\rangle, \left|-\right\rangle, \left|+_y\right\rangle, \left|-_y\right\rangle\}$ to a state $\left|\mathtt{out}\right\rangle = \left|+_{L\cdot\frac{\pi}{4}}\right\rangle$, 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:

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

[8] Here it is worth pointing out that a similar result (in a more complicated method) was achieved in [19]. That technique however, is applied in the information theoretic setting.

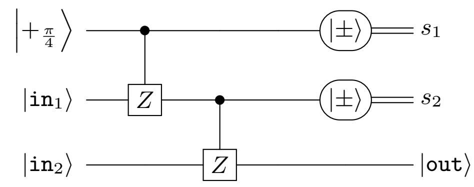

Fig. 2. Merge Gadget

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* $L = L_1L_2L_3 \in \{0,1\}^3$.

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* $S = \{\left|+_{\pi L/4}\right\rangle\}_{L\in\{0,\ldots,7\}} = \{|+\rangle, \left|+_{\frac{\pi}{4}}\right\rangle, .., \left|+_{\frac{7\pi}{4}}\right\rangle\}$.

Sketch Proof (The full proof can be found in the full paper). 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 $poly_2(\cdot)$). $\square$

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 [21] and thus is the basis for classical verification protocols such as that of [33].

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 the full paper) (Fig. 3).

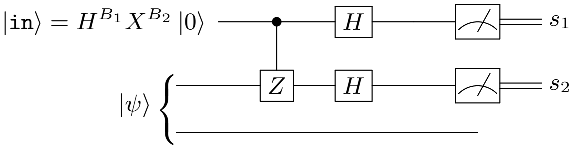

Fig. 3. Blind Measurement Gadget

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 [32] ([7] 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 [25] for a review of this theorem as well as the main proof methods), and other works [52] even extend this lemma to protocols, and also to quantum setting [49, 28]. Unfortunately, these works focus on communication and query complexity and are not really usable in our case.

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, the full proofs being in the full paper:

- 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 $\beta_1, \dots, \beta_t$, 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 [15] 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. 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 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 $t_c = \Omega(n)$.

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 the full paper).

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

Now, we will formalize the previous statements, the proofs are in the full paper.

Conjecture 1 (Yao's XOR Lemma for one-round protocols (classical messages) against quantum adversary).
Let n be the security parameter, $f_n : \mathcal{X}_n \times \mathcal{Y}_n \to \{0,1\}$ be a (possibly non-deterministic) function family (usually not computable in polynomial time), and 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 QPT (in n) 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:

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

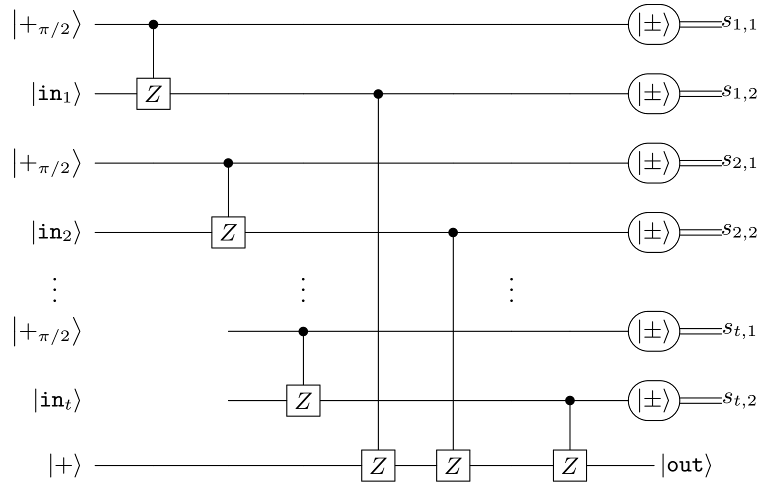

Fig. 4. The XOR gadget circuit $\mathtt{Gad}_\oplus$ (run on server side)

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 $\{|0\rangle, |1\rangle\}$.*

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 $\oplus_{i=1}^{t} b_i$.*

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:

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

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 $|\mathtt{in}_{(i,j)}\rangle$.

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.

Theorem 7 (Malicious-Abort QFactory is correct and secure). *Assuming Conjecture 1, if 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 $\mathcal{A}$:*

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

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 $\Omega(n)$.

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* (Def. 9 and Def. 10). While the 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* [22], the basis-blindness property is not sufficient. What is needed is a stronger property, *verification*, that ensures that the produced state was 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[9]. 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. 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 and furthermore, it is also possible to relax slightly the definition of verifiable QFactory, as it is proven in the full paper.

Definition 11 (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.

[9] 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 [36, 31, 51, 38, 37] 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 the full paper we also 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 [31, 38, 26, 48, 17]. The security proof of the first case can be found in the full paper, while complete 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:

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

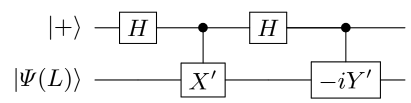

Fig. 5. The isometry of the blind self-testing. Note that the controlled gates are controlled in the X-basis, i.e. $\wedge U_{12}(a\,|+\rangle + b\,|-\rangle)_1 \otimes (|\psi\rangle)_2 = a\,|+\rangle_1 \otimes |\psi\rangle_2 + b\,|-\rangle_1 \otimes U\,|\psi\rangle_2$.

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 [27, 6, 13, 9] 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 11, 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.

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[10] is negligible ϵ':

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

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

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

σ_{junk} is a constant density matrix, ϵ, ϵ' are negligible and $t(\cdot)$ is 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* $\left|+_{L^{(i_k)}\pi/4}\right\rangle \left\langle +_{L^{(i_k)}\pi/4}\right|$. *Thus, 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 prepared correctly. □

Theorem 9 (security). *Protocol 8.2 is a Verifiable QFactory (Definition 11), 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.*

In the full paper we also explain how this task is very similar with self-testing results, and provide the first step for our self-testing result.

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

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

1. S. Aaronson, A. Cojocaru, A. Gheorghiu, and E. Kashefi. On the implausibility of classical client blind quantum computing. *arXiv preprint arXiv:1704.08482*, 2017.
2. D. Aharonov, M. Ben-Or, E. Eban, and U. Mahadev. Interactive Proofs for Quantum Computations. *arXiv e-prints*, page arXiv:1704.04487, Apr 2017.
3. G. Alagic, Y. Dulek, C. Schaffner, and F. 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.
4. F. Armknecht, T. Gagliardoni, S. Katzenbeisser, and A. Peter. General impossibility of group homomorphic encryption in the quantum world. In *International Workshop on Public Key Cryptography*, pages 556–573. Springer, 2014.
5. P. Arrighi and L. Salvail. Blind quantum computation. *International Journal of Quantum Information*, 04, 10 2003.
6. K. AZUMA. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.
7. Z. Brakerski. Quantum FHE (almost) as secure as classical. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 67–95. Springer, 2018.
8. Z. Brakerski, P. Christiano, U. Mahadev, U. V. Vazirani, and T. 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.
9. F. G. Brandao and A. 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.
10. A. Broadbent, J. Fitzsimons, and E. 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.
11. A. Broadbent and S. Jeffery. Quantum homomorphic encryption for circuits of low t-gate complexity. In *Annual Cryptology Conference*, pages 609–629. Springer, 2015.
12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://eprint.iacr.org/2000/067`.
13. C. M. Caves, C. A. Fuchs, and R. Schack. Unknown quantum states: The quantum de finetti representation. *Journal of Mathematical Physics*, 43(9):4537–4559, 2002.

14. A. M. Childs. Secure assisted quantum computation. *Quantum Info. Comput.*, 5(6):456–466, Sept. 2005.
15. A. Cojocaru, L. Colisson, E. Kashefi, and P. Wallden. On the possibility of classical client blind quantum computing. *CoRR*, abs/1802.08759, 2018.
16. A. Cojocaru, L. Colisson, E. Kashefi, and P. Wallden. QFactory: classically-instructed remote secret qubits preparation. *arXiv e-prints*, page arXiv:1904.06303, Apr 2019.
17. A. Coladangelo, A. Grilo, S. Jeffery, and T. Vidick. Verifier-on-a-leash: new schemes for verifiable delegated quantum computation, with quasilinear resources. *arXiv preprint arXiv:1708.07359*, 2017.
18. Y. Dulek, C. Schaffner, and F. Speelman. Quantum homomorphic encryption for polynomial-sized circuits. In *Annual Cryptology Conference*, pages 3–32. Springer, 2016.
19. V. Dunjko and E. Kashefi. Blind quantum computing with two almost identical states. *arXiv e-prints*, page arXiv:1604.01586, Apr. 2016.
20. V. Dunjko, E. Kashefi, and A. Leverrier. Blind quantum computing with weak coherent pulses. *Physical Review Letters*, 108:200502, 08 2011.
21. J. F. Fitzsimons, M. Hajdusek, and T. Morimae. Post hoc verification of quantum computation. *Phys. Rev. Lett.*, 120:040501, Jan 2018.
22. J. F. Fitzsimons and E. Kashefi. Unconditionally verifiable blind quantum computation. *Phys. Rev. A*, 96:012303, Jul 2017.
23. A. Gheorghiu and T. Vidick. Computationally-secure and composable remote state preparation. *arXiv e-prints*, page arXiv:1904.06320, Apr 2019.
24. V. Giovannetti, L. Maccone, T. Morimae, and T. G. Rudolph. Efficient universal blind quantum computation. *Phys. Rev. Lett.*, 111:230501, Dec 2013.
25. O. Goldreich, N. Nisan, and A. Wigderson. *On Yao's XOR-Lemma*, pages 273–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
26. T. Haur Yang and M. Navascus. Robust self testing of unknown quantum systems into any entangled two-qubit states. *Physical Review A*, 87, 10 2012.
27. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
28. H. Klauck, R. Spalek, and R. de Wolf. Quantum and Classical Strong Direct Product Theorems and Optimal Time-Space Tradeoffs. *arXiv e-prints*, pages quant–ph/0402123, Feb. 2004.
29. C.-Y. Lai and K.-M. Chung. On statistically-secure quantum homomorphic encryption. *arXiv preprint arXiv:1705.00139*, 2017.
30. M. Liang. Quantum fully homomorphic encryption scheme based on universal quantum circuit. *Quantum Information Processing*, 14(8):2749–2759, 2015.
31. F. Magniez, D. Mayers, M. Mosca, and H. Ollivier. Self-testing of quantum circuits. 01 2006.
32. U. Mahadev. Classical homomorphic encryption for quantum circuits. In *FOCS*, pages 332–338. IEEE Computer Society, 2018.
33. U. Mahadev. Classical verification of quantum computations. In *FOCS*, pages 259–267. IEEE Computer Society, 2018.
34. A. Mantri, C. A. Pérez-Delgado, and J. F. Fitzsimons. Optimal blind quantum computation. *Physical review letters*, 111(23):230502, 2013.
35. U. Maurer and R. Renner. Abstract cryptography. In *IN INNOVATIONS IN COMPUTER SCIENCE*. Tsinghua University Press, 2011.
36. D. Mayers and A. Yao. Self testing quantum apparatus. *Quantum information & computation*, 4:273, 08 2003.

37. M. 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.
38. M. Mckague, T. Haur Yang, and V. Scarani. Robust self testing of the singlet. *Journal of Physics A: Mathematical and Theoretical*, 45, 03 2012.
39. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
40. T. Morimae, V. Dunjko, and E. Kashefi. Ground state blind quantum computation on aklt state. *Quantum Info. Comput.*, 15(3-4):200–234, Mar. 2015.
41. T. Morimae and K. Fujii. Blind topological measurement-based quantum computation. *Nature communications*, 3:1036, 09 2012.
42. M. Newman and Y. Shi. Limitations on transversal computation through quantum homomorphic encryption. *arXiv preprint arXiv:1704.07798*, 2017.
43. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010.
44. Y. Ouyang, S.-H. Tan, and J. Fitzsimons. Quantum homomorphic encryption from quantum codes. *arXiv preprint arXiv:1508.00938*, 2015.
45. C. 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.
46. J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, Aug. 2018.
47. O. 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.
48. B. W. Reichardt, F. Unger, and U. 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.
49. A. A. Sherstov. Strong direct product theorems for quantum communication and query complexity. *arXiv e-prints*, page arXiv:1011.4935, Nov. 2010.
50. S.-H. Tan, J. A. Kettlewell, Y. Ouyang, L. Chen, and J. F. Fitzsimons. A quantum approach to homomorphic encryption. *Scientific reports*, 6:33467, 2016.
51. 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.
52. E. Viola and A. Wigderson. Norms, xor lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4(7):137–168, 2008.
53. S. Wehner, D. Elkouss, and R. Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412):303, 2018.
54. L. Yu, C. A. Pérez-Delgado, and J. F. Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Physical Review A*, 90(5):050303, 2014.