On the Possibility of Classical Client Blind Quantum Computing

cryptography

MDPI

Article

On the Possibility of Classical Client Blind Quantum Computing

Alexandru Cojocaru [1,*], Léo Colisson [2,*], Elham Kashefi [1,2] and Petros Wallden [1]

[1] School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK; ekashefi@exseed.ed.ac.uk (E.K.); petros.wallden@ed.ac.uk (P.W.)
[2] Département Informatique et Réseaux, LIP6, Sorbonne Université, 75005 Paris, France
* Correspondence: a.d.cojocaru@sms.ed.ac.uk (A.C.); leo.colisson@lip6.fr (L.C.)

Abstract: Classical client remote state preparation (CC − RSP) is a primitive where a fully classical party (client) can instruct the preparation of a sequence of random quantum states on some distant party (server) in a way that the description is known to the client but remains hidden from the server. This primitive has many applications, most prominently, it makes blind quantum computing possible for classical clients. In this work, we give a protocol for classical client remote state preparation, that requires minimal resources. The protocol is proven secure against honest-but-curious servers and any malicious third party in a game-based security framework. We provide an instantiation of a trapdoor (approximately) 2-regular family of functions whose security is based on the hardness of the Learning-With-Errors problem, including a first analysis of the set of usable parameters. We also run an experimentation on IBM's quantum cloud using a toy function. This is the first proof-of-principle experiment of classical client remote state preparation.

Keywords: remote state preparation; blind quantum computing; learning with errors

Citation: Cojocaru, A.; Colisson, L.; Kashefi, E.; Wallden, P. On the Possibility of Classical Client Blind Quantum Computing. *Cryptography* **2021**, *5*, 3. https://doi.org/10.3390/cryptography5010003

Received: 22 December 2020
Accepted: 15 January 2021
Published: 24 January 2021

Contents

1. Introduction and Related Works

The recent interest in quantum technologies has brought forward a vision of *quantum internet* [1] that could implement a collection of known protocols for enhanced security or communication complexity [2]. On the other hand, the rapid development of quantum hardware has increased the computational capacity of quantum servers that could be linked in such a communicating network. This raised the necessity and importance of privacy-preserving functionalities such as the research developed around quantum computing on encrypted data (see a recent review in [3]).

However, there exist some challenges in adapting widely the above vision: A reliable long-distance quantum communication network connecting all the interested parties might be very costly and hard to realize. Moreover, currently, some of the most promising quantum computation devices (e.g., superconducting such as the devices developed by IBM, Google) do not yet offer the possibility of "networked" architecture, i.e., cannot receive and send quantum states.

Therefore, given the crucial role of quantum cloud services, the most important challenge imposing practicality limitations refers to the need of quantum communication. On the one hand, quantum communication is required for the most secure quantum computing primitives, but also for the development of extended quantum communication networks reaching all interested parties (even those with light devices) that need to ensure the compatibility between different quantum communication and computing platforms. For this reason, there has been extensive research focusing on the practicality aspect of quantum delegated computation protocols (and related functionalities). One direction is to reduce the required communications by exploiting classical fully-homomorphic-encryption schemes [4–6], or by defining their direct quantum analogues [7–10]. Different encodings, on the client-side, could also reduce the communication [11,12]. 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 [13–16], completing this picture remains an open problem.

Another very important direction is to consider fully-classical client protocols, compatible with the no-go results, that can therefore achieve more restricted levels of security. The first such procedure achieving statistical security (but not for universal computations) was proposed in [17].

In recent breakthrough results of Mahadev [18], a universal blind delegated protocol for a fully-classical client was proposed under post-quantum computational security. This work, as well as the result of Brakerski [19] which improves the security of [18], proposed quantum fully homomorphic encryption schemes for quantum computations.

The solution we propose in this work , takes a different modular approach. We replace the need for a quantum communication with a classical primitive mimicking a quantum channel. Specifically, what we define and achieve is a version of "classical client remote state preparation" ($\mathsf{CC-RSP}$) formally defined in [20], where the security is post-quantum in the honest-but-curious model.

Note: We first introduced this primitive and the protocol presented here, in a preprint in 2018 that was also presented in QCrypt '18, which appeared in between the above mentioned works, but was not published. Here we present an extended version of that work, where we have extended our initial work with a series of results which include: the formal proof of security in a game-based framework and realising the first proof-of-principle experiment on a quantum cloud service of such a primitive using a small-scale toy function.

1.1. Related Work

The first proposal for a fully-classical delegated quantum computation protocol appeared in [18], followed by another classical fully homomorphic encryption scheme for quantum computation with improved security parameters in [19]. However, unlike their solution, our approach is modular, as the primitive we propose can be used for a number

of functionalities, including delegation of quantum computations (such as [21]), to replace the required quantum communication and make the functionalities directly accessible to classical parties.

Following the introduction of this family of primitives, known as classical client remote state preparation ($\mathsf{CC}-\mathsf{RSP}$), in our earlier work [22] (version 2), more works have arose that present constructions of $\mathsf{CC}-\mathsf{RSP}$ achieving security against arbitrarily deviating, computationally bounded, adversaries. In [23], it is provided a $\mathsf{CC}-\mathsf{RSP}$ protocol, whose security holds against any malicious server. Concurrently in [24], the authors provide a verifiable version of this $\mathsf{CC}-\mathsf{RSP}$ primitive with composable security, but under some assumption known as "measurement buffer". A thorough analysis for the security of this important class of resources was given in [20], where it is shown that the family of $\mathsf{CC}-\mathsf{RSP}$ cannot be composably-secure unless they leak the description of the produced quantum state to the adversary. In the recent work of [25], the author proposes a delegated quantum computation protocol in which the quantum communication is independent of the size of the computation and depends only (polynomially) on the security parameter (and where the size of the quantum circuit the client has to perform, is also polynomial in the security parameter). This property referred here as "succinct" complexity for the client is achieved in a protocol whose security is proven in the quantum random oracle model. For more details on the general topic of quantum cryptography there are numerous reviews (see for example [26,27]).

1.2. Our Contributions

1. We define the primitive *classical client remote state preparation* ($\mathsf{CC}-\mathsf{RSP}_\theta$) in Section 3. In the earlier version of this work we called this primitive *secret random qubit generator*, but we switched to the term remote state preparation (RSP), which is the terminology established by the quantum cryptography community. The parameter θ refers to the set of quantum states produced by the primitive, which are the quantum states $\{|+_\theta\rangle\}_{\theta\in\{0,\cdots,7\pi/4\}}$. $\mathsf{CC}-\mathsf{RSP}_\theta$ can replace the need for quantum channel between parties in certain quantum communication protocols with the trade-off that the protocols become computationally secure (against *quantum* adversaries).
2. We give a basic protocol ($\mathsf{HBC}-\mathsf{QFactory}$) that achieves this functionality from a correctness point of view, given a trapdoor one-way function that is quantum-safe, two-regular and collision resistant in Section 4 and prove its correctness.
3. We prove the security of the $\mathsf{HBC}-\mathsf{QFactory}$ against honest-but-curious server (server follows the protocol specifications, but can try to infer any information about the secret from the classical transcripts) or against any malicious third party using a game-based security definition. To show the security, we prove that the classical description of the generated qubits is a hard-core function (following a reduction similar to that of the Goldreich–Levin Theorem) in Section 5.
4. While the above-mentioned results do not depend on the specific function used, the existence of such specific functions (with all desired properties) makes the $\mathsf{CC}-\mathsf{RSP}_\theta$ a practical primitive that can be employed as described in this paper. In Section 6, we first give methods for obtaining two-regular trapdoor one-way functions with extra properties (collision resistant or second preimage resistant) assuming the existence of simpler trapdoor one-way functions (permutation trapdoor or homomorphic, injective trapdoor functions). We use reductions to prove that the resulting functions maintain all the properties required. Furthermore, we give in Section 6.3 an explicit family of functions that respect all the required properties based on the security of the Learning-With-Errors problem as well as a possible instantiation of the parameters. This function is also quantum-safe, and thus directly applicable for our setting. Note, that other functions may also be used, such as the one in [28] or functions based on the Niederreither cryptosystem and the construction in [29].
5. Finally, we implement $\mathsf{HBC}-\mathsf{QFactory}$ on the quantum computer IBM Quantum Experience using a toy function (given the current limited number of available qubits

we consider a 2-regular function acting on a small number of bits, consequently, it cannot be post-quantum secure). Hence, we provide in addition to the theoretical results, an experimental evidence of the correctness and output distribution of the $\mathsf{HBC}-\mathsf{QFactory}$ protocol on a real quantum device. This is the first implementation of an $\mathsf{RSP}-\mathsf{CC}$ protocol on a quantum cloud service.

1.3. Applications

The $\mathsf{CC}-\mathsf{RSP}_\theta$ primitive, viewed as a resource, has a wide range of applications. Here we give a general overview of the applications, while for details on *how* to use the *exact output* of the $\mathsf{CC}-\mathsf{RSP}_\theta$ obtained in this paper in specific protocols, we refer the reader to Appendix A. $\mathsf{CC}-\mathsf{RSP}_\theta$ enables fully-classical parties to participate in many quantum protocols using only public classical channels and a single (potentially malicious) quantum server.

The first type of applications concerns a large class of delegated quantum computation protocols, including blind quantum computation and *verifiable* blind quantum computation. These protocols are of great importance, enabling information-theoretically secure (and verifiable) access to a quantum cloud. However, the requirement for quantum communication limits their domain of applicability. This limitation is removed by replacing the off-line preparation stage with our $\mathsf{HBC}-\mathsf{QFactory}$ protocol. Concretely, we can use $\mathsf{HBC}-\mathsf{QFactory}$ to implement the blind quantum computation protocol of [21], as well as the *verifiable* blind quantum computation protocols (e.g., those in [30–32]), in order to achieve classical client secure and verifiable access to a quantum cloud.

In all these cases, the cost of using $\mathsf{CC}-\mathsf{RSP}_\theta$ is that the security becomes post-quantum computational (from information-theoretic). However, the possibility of information-theoretically secure classical client blind quantum computation seems highly unlikely due to strong complexity-theoretic arguments given in [15] and therefore this is the best we could hope for.

The second type of applications involves a more general family of protocols for which their quantum communication consists of random single qubits similar to those provided by the QFactory, such as: quantum-key-distribution [33], quantum money [34], quantum coin-flipping [35], quantum signatures [36], etc.

In the two previous families of applications, QFactory can be used in order to replace quantum clients with classical clients. There is **a third type of applications** of QFactory that is quite novel: QFactory could be used in order to improve the efficiency and security of some protocols. More specifically, it could be used in protocols where each party needs to make sure that all the other parties are honest. Three major examples of such applications would be the following: two-party quantum computation [37,38], multiparty quantum computation [39] and quantum one-time programs [40]. Indeed, most of these protocols suffer from the fact that one party could try to cheat by sending malicious states instead of the honest single qubit states. Using QFactory, all communications between the client(s) and the server become classical and this observation can be exploited to use classical techniques (e.g., zero-knowledge proofs) to achieve exponential security. Specifically, because of this issue, ref. [39] is proven secure only when there is no dishonest coalition between some clients and the server, while some other works like [38] try to resolve this issue by using a "cut-and-choose" technique where the client sends s qubits, as well as s commitments to their corresponding classical descriptions. All but one commitment are opened for testing and the remaining qubit is then used in the actual computation, leading to a security level of $\frac{1}{s}$, i.e., security that can grow only linearly with the security parameter. QFactory can improve this in the following way. To make sure that the client(s) behave honestly, the client(s) are requested to send a (post-quantum, but classical) zero-knowledge proof, proving that all the data were generated honestly (i.e., that he knows the trapdoor corresponding to the public description k of the function, as well as the two preimages corresponding to the image y given by the server (see later). One could also add a distributed coin toss before the protocol (where the randomness provided by the client is

not revealed, but used in the zero-knowledge proofs) to make sure that the randomness used by the client is not chosen in a malicious way). In this way, QFactory, can achieve an exponential security in the protocol of [38], and also improve the security proof of other protocols. This approach was recently used in the setting of two-party quantum computation [41].

1.4. Overview of the Protocol and Proof

The general idea is that a classical client gives instructions to a quantum server to perform certain actions (quantum computation). Those actions lead to the server having as output a single qubit, which is randomly chosen from within a set of possible states of the form $|0\rangle + e^{ir\pi/4}|1\rangle$, where $r \in \{0, \cdots, 7\}$. The randomness of the output qubit is due to the (fundamental) randomness of quantum measurements that are part of the instructions that the client gives. Moreover, the server cannot guess the value of r any better than if he had just received that state directly from the client (up to negligible probability). This is possible because the instructed quantum computation is generically a computation that is hard to (i) classically simulate and (ii) to reproduce quantumly because it is unlikely (exponentially in the number of measurements) that by running the same instructions the server obtains the exact same measurement outcomes twice. On the other hand, we wish the client to *know* the classical description and thus the value of r. To achieve this task, the instructions/quantum computation the client uses are based on a family of trapdoor one-way functions with certain extra properties (The functions should also be two-regular (each image has exactly two preimages), quantum safe (secure against quantum attackers) and collision resistant (hard to find two inputs with the same image)). Such functions are hard to invert (e.g., for the server) unless someone (the client in our case) has some extra "trapdoor" information t_k. This extra information makes the quantum computation easy to classically reproduce for the client, which can recover the value r, while it is still hard to classically reproduce for the server. Sending random qubits of the above type, is exactly what is required from the client in most of the protocols and applications given earlier, while with simple modifications our protocol could achieve other similar sets of states.

Our $\mathsf{HBC-QFactory}$ protocol can heuristically be described in the next steps:
Preparation. The client randomly selects a function f_k, from a family of trapdoor one-way, quantum-safe, two-regular and collision resistant functions. The choice of f_k is public (server knows), but the trapdoor information t_k needed to invert the function is known only to the client.

Stage 1: Preimages Superposition. The client instructs the server (i) to apply Hadamard(s) on the control register, (ii) to apply U_{f_k} on the target register i.e., to obtain $\sum_x |x\rangle \otimes |f_k(x)\rangle$ and (iii) to measure the target register in the computational basis, in order to obtain a value y. This collapses his state to the state $(|x\rangle + |x'\rangle) \otimes |y\rangle$, where x, x' are the unique two preimages of y.

Remarks. First we note that each image y appears with same probability (therefore, obtaining twice the same y happens with negligible probability). We now consider the first register $|x\rangle + |x'\rangle = |x_1 \cdots x_n\rangle + |x'_1 \cdots x'_n\rangle$, where the subscripts denote the different bits of the corresponding preimages x and x'. We rewrite this:

$$\left(\otimes_{i\in\bar{G}} |x_i\rangle\right) \otimes \left(\prod_{j\in G} X^{x_j}\right)\left(|0\cdots 0\rangle_G + |1\cdots 1\rangle_G\right)$$

where $\bar{G}$ is the set of bits positions where x, x' are identical, G is the set of bits positions where the preimages differ, while we have suitably changed the order of writing the qubits. It is now evident that the state at the end of Stage 1 is a tensor product of isolated $|0\rangle$ and $|1\rangle$ states, and a Greenberger–Horne–Zeilinger (GHZ) state with random X's applied. The crucial observation is that the connectivity (which qubit belongs to the GHZ and which

doesn't) depends on the XOR of the two preimages $x \oplus x'$ and is computationally impossible to determine, with non-negligible advantage, without the trapdoor information t_k.

Stage 2: Squeezing. The client instructs the server to measure each qubit i (except the output) in a random basis $\{|0\rangle \pm e^{i\alpha_i \pi/4}|1\rangle\}$ and return back the measurement outcome b_i. The output qubit is of the form $|+_\theta\rangle = |0\rangle + e^{i\theta}|1\rangle$, where:

$$\theta = \frac{\pi}{4}(-1)^{x_n} \sum_{i=1}^{n-1} (x_i - x'_i)(4b_i + \alpha_i) \bmod 8 \tag{1}$$

Intuitively, measuring qubits that are not connected has no effect to the output, while measuring qubits within the GHZ part, rotates the phase of the output qubit (by a $(-(1)^{x_i}\alpha_i + 4b_i)\pi/4$ angle). The above intuition shows that our $\mathsf{HBC-QFactory}$ protocol is correct, as fully proven in Theorem 5.

Security. The protocol is secure, if we can prove that the server (or other third parties) cannot guess (obtain noticeable advantage in guessing) the classical description of the state, i.e., the value of θ. We consider a quantum-honest-but-curious server which means that he essentially follows the protocol and the security reduces to proving that the server cannot use his classical information to obtain any advantage in guessing the classical description of the (honest) quantum output.

The server does not know the two preimages x, x' and needs to guess θ (which is a three-bit string) from the value of the image y. The key technical part of the security proof is showing a variant of the Goldreich–Levin theorem [42], that (informally) states that the predicate represented by the inner product of the preimage of a one-way function with a random vector, taken modulo 2, is indistinguishable from a random bit. In our case, θ has a similar expression (1) as it can be expressed as the inner product between the XOR of two preimages and a random vector taken modulo 8. We prove in Theorem 7 that if a computationally bounded server could obtain non-trivial advantage in guessing θ, then he could also break the property of "second preimage resistance" which we requested for our function f_k. Note, that in our protocol, we actually request the strongest collision-resistance property that implies the second preimage resistance. To prove this theorem we first express each of the 3 bits of θ as an XOR between a Goldreich–Levin type of predicate and some extra functions. Each of these predicates, instead of having a preimage in the inner product, they have the (bitwise) XOR of the two preimages. We therefore show that guessing any of those predicates would break the (stronger) assumption of collision resistance, reaching a contradiction. Then, to connect the hardness of computing the bits of θ (each of the three predicates) with the hardness of computing θ, we use the result of [43] to address the issue of possible correlations. The technical hardest part of Theorem 7 is on the one hand, that we fix all but one variable in the expression of each predicate (bit of θ), with an extra cost that is an inverse polynomial probability and on the other hand, that we then use a "disentangling" trick to express the bits as the XOR between a Goldreich–Levin predicate and an extra function (now independent of the other variables).

Using the property of θ mentioned above, we then prove the full security of the $\mathsf{HBC-QFactory}$ protocol by proving the game-based security holds by a reduction to the hardcore function property of θ in Theorem 6. More specifically, we show that if the server runs honestly the protocol, but keeps a record of the classical transcript of the protocol together with the measurement outcomes he performs, then it is hard for him to correlate this internal view of the protocol with the protocol output $(\theta, |+_\theta\rangle)$.

The function. Our protocol relies on using functions that have a number of properties (one-way, trapdoor, two-regular, collision resistant (see Remark 1), quantum safe). Any function satisfying those conditions is suitable for our protocol. While in first thought some of these appear hard to satisfy jointly (e.g., two-regularity and collision resistance), we give two constructions that achieve those properties from simpler functions: one from

injective, homomorphic trapdoor one-way function and one from bijective trapdoor one-way function. Both constructions define a new function that has domain extended by one bit, and the value of that bit "decides" whether one uses the initial basic function or not.

More specifically, for the *first construction*, let us denote the injective, homomorphic, trapdoor one-way function by g_k, with k the public description of the function and t_k the trapdoor—used for the inversion of the function g_k. Then, we pick at random an element x_0 from the domain of g_k. The public description k' of the new desired function f will be k along with $g_k(x_0)$ and the corresponding trapdoor $t_{k'}$ of f would be t_k along with x_0.

Then, the function f, which is evaluated by the server, is described as: $f_{k'}(x, c) = g_k(x) + c \cdot g_k(x_0)$, which due to the homomorphic property of g, can be rewritten as: $f_{k'}(x, c) = g_k(x + c \cdot x_0)$. Now, we can see the 2-regularity property of f as, since g_k is injective, f will always have exactly 2 preimages of the form: x and $x + x_0$, which can always be efficiently computed from the image of f using $t_{k'}$. The one-wayness and quantum-safety of f are then proved by reduction to the one-wayness, respectively quantum-safety of g and finally, we prove the collision-resistance of f, by reducing it to the one-wayness of g.

For the *second construction*, we denote again by g, the bijective, trapdoor one-way function. Then, in order to construct f, we will basically use 2 such functions g: the public description k' of f will consist of k_0 and k_1—the public descriptions of g_{k_0} and g_{k_1} and the trapdoor of $f_{k'}$ will consist of the pair $t_{k'} = (t_{k_0}, t_{k_1})$—the trapdoors of g_{k_0} and g_{k_1}.

Then, the function f, evaluated by the server, is described as: $f_{k'}(x, c) = g_{k_c}(x)$. Now, we can see the 2-regularity property of f as, since g_k is bijective, every y from the image of f, will have 2 preimages, namely the unique preimage of g_{k_0} and the unique preimage of g_{k_1}, which can be both computed from y using $t_{k'}$. Then, in Appendix C we prove the one-wayness and quantum-safety of f by reduction to one-wayness and quantum-safety of g and finally, we prove the second preimage-resistance of f by reduction to the one-wayness of one of the 2 functions g.

We then give a real instantiation of the required function f based on the first type of construction, starting from the injective trapdoor one-way function defined in [44], which is derived from the Learning-With-Errors problem: $g_K(s, e) = s^t K + e^t$, where $s \in \mathbb{Z}_q^n$ and $K \in \mathbb{Z}_q^{n \times m}$, so using the above notation, we have $x = (s, e)$ and $x_0 = (s_0, e_0)$. This function also seems to satisfy the homomorphic property with respect to addition modulo q: $g_K(s, e) + g_K(s_0, e_0) \bmod q = (s^t K + e^t + s_0^t K + e_0^t) \bmod q = g_K((s + s_0) \bmod q, e + e_0)$. Unfortunately, things are not so simple, because the domain of the error vector $e \in \mathbb{Z}^m$ is such that each component of e is bounded by some value μ, in order for g to be injective and correctly inverted using the trapdoor. This implies that g is homomorphic as long as $e + e_0$ is also bounded (in infinite norm) by μ. In our case, this means that $e + e_0$ may not be small enough to lie within g's domain, so it may be possible to have only one preimage for some image y. To overpass these problems, we do the following:

When we are constructing the trapdoor for the function f, in particular when we are sampling $x_0 = (s_0, e_0)$ from the domain of g, we will in fact sample e_0 from a smaller set, such that when it will be added together with a random input e, the total noise vector will still be small enough to lie within the domain of g with some good probability.

What we prove is that as long as e_0 is sampled from a subset of the domain of g such that e_0 is now bounded by $\mu' = \frac{\mu}{m}$, we will get that with at least a constant probability, $e + e_0$ is inside the domain of g, or in other words that f is now 2-regular with at least a constant probability. What remains to be proven is that when e_0 is restricted to this smaller domain, $g_K(s_0, e_0)$ still cannot be inverted by an adversary. Therefore, as a final step we prove that there exists an explicit choice of parameters such that both g and the restriction of g to the domain of e_0 are one-way functions and such that all the other properties of g are preserved.

As a result, under this choice of parameters, we obtain our desired function f, satisfying all required properties: one-wayness, trapdoor, collision-resistance and 2-regularity (with a least a constant probability), where the hardness of f is proven by reduction to

worst-case hardness of approximating short vectors problems, with polynomial approximation factor, which is the current standard in lattice-based cryptosystems.

Remark 1. *It appears that the second preimage resistance property will be enough to prove the security of our scheme in the honest-but-curious setting. However, as soon as the server can be malicious, the collision resistance property will be very important, else the server might forge known valid states, which would break the security.*

2. Preliminaries

2.1. Classical Definitions

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 D the domain of the functions, while $D(n)$ is the subset of strings of length n.

Definition 1 (Quantum-Safe (informal))**.** *A protocol/function is quantum-safe (also known as post-quantum secure), if all its properties remain valid when the adversaries are QPT (instead of PPT).*

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 2 (One-way)**.** *A family of functions* $\{f_k : D \to R\}_{k \in K}$ *is* ***one-way*** *if:*

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

Definition 3 (Second preimage resistant)**.** *A family of functions* $\{f_k : D \to R\}_{k \in K}$ *is* ***second preimage resistant*** *if:*

- *There exists a PPT algorithm that can compute* $f_k(x)$ *for any index function* k*, outcome of the PPT parameter-generation algorithm Gen and any input* $x \in D$*;*
- *for any PPT algorithm* $\mathcal{A}$*, given an input* x*, it can find a different input* x' *such that* $f_k(x) = f_k(x')$ *with at most negligible probability over the choice of* k*:*
  $$\Pr_{\substack{k \leftarrow Gen(1^n) \\ x \leftarrow D \\ rc \leftarrow \{0,1\}^*}} [\mathcal{A}(k, x) = x' \text{ such that } x \neq x' \text{ and } f_k(x) = f_k(x')] \leq \mathsf{negl}(n)$$
  where rc is the randomness of $\mathcal{A}$*;*

Definition 4 (Collision resistant)**.** *A family of functions* $\{f_k : D \to R\}_{k \in K}$ *is* ***collision resistant*** *if:*

- *There exists a PPT algorithm that can compute* $f_k(x)$ *for any index function* k*, outcome of the PPT parameter-generation algorithm Gen and any input* $x \in D$*;*
- *any PPT 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).*

Theorem 1. *[45] Any function that is collision resistant is also second preimage resistant.*

Definition 5 (k-regular). *A deterministic function $f: D \to R$ is **k-regular** if $\forall y \in \mathrm{Im}(f)$, we have $|f^{-1}(y)| = k$.*

Definition 6 (Trapdoor Function). *A family of functions $\{f_k : D \to R\}$ 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;*
- *$\{f_k : D \to R\}_{k \in K}$ 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) with non-negligible probability over the choice of (k, t_k) and uniform choice of x. Note, that 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 trapdoor function where the inversion procedure returns both preimages for any function output.*

Definition 7 (Hard-core Predicate). *A function $hc: D \to \{0,1\}$ is a **hard-core predicate** for a function f if:*

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

Definition 8 (Hard-core Function). *A function $h : D \to E$ is a **hard-core function** for a function f if:*

- *There exists a PPT algorithm that can compute $h(x)$ for any input x;*
- *for any PPT algorithm $\mathcal{A}$ when given $f(x)$, $\mathcal{A}$ can distinguish between $h(x)$ and a uniformly distributed element in E with at most negligible probability:*

$$\Big| \Pr_{x \leftarrow D(n)} [\mathcal{A}(f(x), h(x)) = 1] - \Pr_{\substack{x \leftarrow D(n) \\ r \leftarrow E(|h(x)|)}} [\mathcal{A}(f(x), r) = 1] \Big| \leq \mathsf{negl}(n)$$

The intuition behind this definition is that as far as a QPT adversary is concerned, the hard-core function appears indistinguishable from a randomly chosen element of the same length.

Theorem 2 (Goldreich–Levin [42]). *From any one-way function $f: D \to R$, we can construct another one-way function $g: D \times D \to R \times D$ and a hard-core predicate for g. If f is a one-way function, then:*

- *$g(x, r) = (f(x), r)$ is a one-way function, where $|x| = |r|$.*
- *$hc(x, r) = \langle x, r \rangle \bmod 2$ is a hard-core predicate for g.*

Informally, the Goldreich–Levin theorem is proving that when f is a one-way function, then $f(x)$ is hiding the xor of a random subset of bits of x from any PPT adversary. The Goldreich–Levin proof is using a reduction from breaking the hard-core predicate $hc(x, r)$ to breaking the one-wayness of g. In this paper, the functions we consider are one-way against quantum adversaries, and using the same reduction we conclude that $hc(x, r)$ is a hard-core predicate against QPT adversaries.

Theorem 3 (Vazirani–Vazirani XOR-Condition Theorem [43]). *Function h is hard-core function for f if and only if the xor of any non-empty subset of h's bits is a hard-core predicate for f.*

The Learning-With-Errors problem (LWE) can be described in the following way:

Definition 9 (LWE problem (informal))**.** *Given s, an n dimensional vector with elements in $\mathbb{Z}_q$, the task is to distinguish between a set of polynomially many noisy random linear combinations of the elements of s and a set of polynomially many random numbers from $\mathbb{Z}_q$.*

Regev [46] and Peikert [47] 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, problems which are conjectured to be hard even for quantum computers.

Theorem 4 (Reduction LWE, from ([46], 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.*

2.2. Quantum Definitions

We assume basic familiarity with quantum computing notions. For any function $f : A \to B$ that can be described by a polynomially-sized classical circuit, we define the controlled-unitary U_f, as acting in the following way:

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \;\; \forall x \in A \;\; \forall y \in B, \tag{2}$$

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

3. $\mathsf{CC-RSP}_\theta$ Primitive

In many distributed protocols the required communication consists of sending sequence of single qubits prepared in random states that are unknown to the receiver (and any other third parties). What we want to achieve through our $\mathsf{CC-RSP}_\theta$ primitive (Algorithm 1) is a way to generate remotely single qubits that are random and (appear to be) unknown to all parties but the "client" that gives the instructions.

Definition 10. *Let $|+_\theta\rangle = 1/\sqrt{2}(|0\rangle + e^{i\theta}|1\rangle)$. We define the set of states*

$$R := \{|+_\theta\rangle\} \;\; \textit{where} \;\; \theta \in \{0, \pi/4, \pi/2, \cdots, 7\pi/4\} \tag{3}$$

By including magic states ($|+_{\pi/4}\rangle$), this set of states can be viewed as a "universal" resource, as applying Clifford operations on those states is sufficient for universal quantum computation. Furthermore, it is sufficient to implement both blind quantum computation (e.g., [21]) and verifiable blind quantum computation (e.g., [32]).

We emphasize that the aim of defining an ideal functionality is to highlight the task we want to achieve (in terms of correctness) with our protocol $\mathsf{HBC-QFactory}$, rather than using it in a simulation or composable security definition.

Remarks: (i) The outcome of this primitive is the client "sending" the qubit $|+_\theta\rangle$ (that he knows) to the server, thus simulating a quantum channel. (ii) We note that there is an abort possibility and some auxiliary classical messages (m_C, m_S), both included to make the primitive general enough to allow for our construction. Furthermore, the classical description of the qubit, θ, and the classical messages (m_C, m_S) are totally uncorrelated (as θ is chosen independently at random for each (m_C, m_S). (iii) While the server can learn something about the classical description (e.g., by measuring the qubit), this information is limited and is the exact same information that he could obtain if the client had prepared and sent him a random qubit.

Algorithm 1 Primitive: Classical Channel Remote State Preparation ($\mathsf{CC}-\mathsf{RSP}_\theta$)

Requirements: Client is a purely classical party with no access to quantum resources.
Public Information: A distribution on pairs of lists M, intuitively containing the values of the classical variables used by the client and by the server.
Trusted Party:
– With some probability p returns to both parties abort, otherwise:
– Samples $(m_C, m_S) \leftarrow M$
– Samples $\theta \leftarrow \{0,1\}^3 \cdot \frac{\pi}{4}$
– Prepares a qubit in state $|+_\theta\rangle$
Outputs:
– Either returns abort to both client and server
– Or returns (m_C, θ) to the client, and $(m_S, |+_\theta\rangle)$ to the server

4. The Real Protocol

To construct our protocol $\mathsf{HBC}-\mathsf{QFactory}$ (Algorithm 2) we assume the existence of a family $\{f_k\colon \{0,1\}^n \to \{0,1\}^m\}_{k\in K}$ of trapdoor one-way functions that are two-regular and collision resistant (or the weaker second preimage resistance property, see Remark 1) even against a quantum adversary. For any y, we will denote by $x(y)$ and $x'(y)$ the two unique different preimages of y by f_k (if the y is clear, we may remove it). Note that because of the two-regularity property $m \geq n-1$. We use subscripts to denote the different bits of the strings. See Section 6 for our function. With that choice, we are guaranteed that the last bits of the two preimages are always different, and thus no need for an abort. We keep the protocol general so that different functions can be used.

Algorithm 2 Real $\mathsf{HBC}-\mathsf{QFactory}$ Protocol

Requirements:
Public: A family $\mathcal{F} = \{f_k : \{0,1\}^n \to \{0,1\}^m\}$ of trapdoor one-way functions that are quantum-safe, two-regular and collision resistant (or second preimage resistant, see Remark 1)
Input:
– Client: uniformly samples a set of random three-bits strings $\alpha = (\alpha_1, \cdots, \alpha_{n-1})$ where $\alpha_i \leftarrow \{0,1\}^3$, and runs the algorithm $(k, t_k) \leftarrow \mathrm{Gen}_{\mathcal{F}}(1^n)$. The α and k are public inputs (known to both parties), while t_k is the "private" input of the client.
Stage 1: Preimages superposition
– Client: instructs server to prepare one register at $\otimes^n H|0\rangle$ and second register initiated at $|0\rangle^m$
– Client: sends k to server and the server applies U_{f_k} using the first register as control and the second as target
– Server: measures the second register in the computational basis, obtains the outcome y and returns this result y to the client. Here, an honest server would have a state $(|x\rangle + |x'\rangle) \otimes |y\rangle$ with $f_k(x) = f_k(x') = y$ and $y \in \mathrm{Im}\, f_k$.
Stage 2: Squeezing
– Client: instructs the server to measure all the qubits (except the last one) of the first register in the $\left\{|0\rangle \pm e^{\alpha_i \pi/4}|1\rangle\right\}$ basis. Server obtains the outcomes $b = (b_1, \cdots, b_{n-1})$ and returns the result b to the client
– Client: using the trapdoor t_k computes x, x'. Then check if the n-th bit of x and x' (corresponding to the y received in Stage 1) are the same or different. If they are the same, returns abort, otherwise, obtains the classical description of the server's state.
Output: If the protocol is run honestly, when there is no abort, the state that server has is $|+_\theta\rangle$, where the client (only) knows the classical description (see Theorem 5):

$$\theta = \frac{\pi}{4}(-1)^{x_n} \sum_{i=1}^{n-1} (x_i - x'_i)(4b_i + \alpha_i) \bmod 8 \tag{4}$$

Remarks: The first thing to note is that the server should not only be unable to guess θ from his classical communications, but he should also be unable to distinguish it from a random string with probability greater than negligible. We will prove this later, but for now it is enough to point out that θ depends on the preimages x and x' of y (which the client can obtain using t_k).

The second thing to note is that while our expression of θ resembles the inner product in the Goldreich–Levin (GL) theorem, it differs in a number of places and our proof (that θ is a hard-core function), while it builds on the GL theorem proof, is considerably more complicated. Details can be found in the security proof, but here we simply mention the differences: (i) our case involves three-bits rather than a predicate, and the different bits, if we view them separately, may not be independent, (ii) we have a term $(x - x')$ rather than a single preimage, so rather than the one-way property of the function we will need the second preimage resistance and (iii) for the same reason, if we view our function as an inner product, it can take both negative and positive values ($(x - x')$ could be negative).

A third thing to note is that we have singled-out the last qubit of the first register, as the qubit that will be the output qubit. One could have a more general protocol where the output qubit is chosen randomly, or, for example, in the set of the qubits that are known to have different bit values between x and x', but this would not improve our analysis so we keep it like this for simplicity. Moreover, while the "inner product" normally involves the full string x that one tries to invert, in our case, it does not include one of the bits (the last) of the string we wish to invert. It is important to note that it does not change anything to our proofs, since if one can invert all of the string apart from one bit with inverse polynomial probability of success, then trivially one can invert the full string with inverse polynomial probability (by randomly guessing the remaining bit or by trying out both values of that bit). Therefore, all the proofs by contradiction are still valid and in the remaining; for notational simplicity, we will take the inner products to involve all n bits.

Correctness and Intuition

Theorem 5. *If both the client and the server follow Algorithm 2, the protocol aborts when $x_n = x'_n$, while otherwise the server ends up with the output (single) qubit being in the state $|+_\theta\rangle$, where θ is given by Equation (4).*

Proof. In the first stage, before the first measurement, but after the application of U_{f_k}, the state is $\sum_x |x\rangle \otimes |f_k(x)\rangle$. What the measurement does, is that it collapses the first register in the equal superposition of the two unique preimages of the measured $y = f_k(x) = f_k(x')$, in other words in the state $(|x\rangle + |x'\rangle) \otimes |y\rangle$. It is not possible, even for malicious adversary (not considered here), to force the output of the measurement to be a given y (see [48] for relation of PostBQP with BQP). This completes the first stage of the protocol. Before proceeding with the proof of correctness we make three observations.

By the second preimage resistance property of the trapdoor function, learning x is not sufficient to learn x' but with negligible probability, and intuitively, by the stronger collision resistance property, even a malicious server cannot forge a state $|x\rangle + |x'\rangle$ (with $f(x) = f(x')$) fully known to him.

Then, we examine what happens if the last bit of x and x' are the same and see why the protocol aborts. In this case, in the first register, the last qubit is in product form with the remaining state, and therefore any further measurements in stage 2 do not affect it, leaving it in the state $|x_n\rangle$. As a result of this, the output state is not of the form of Equation (4), while including this states in the set of possible outputs would change considerably our analysis.

Finally, we should note that the resulting state is essentially a Greenberger–Horne–Zeilinger (GHZ) state [49]: let G be the set of bits positions where x and x' differ (which include n – output qubit), while $\bar{G}$ is the set where they are identical. The state is then

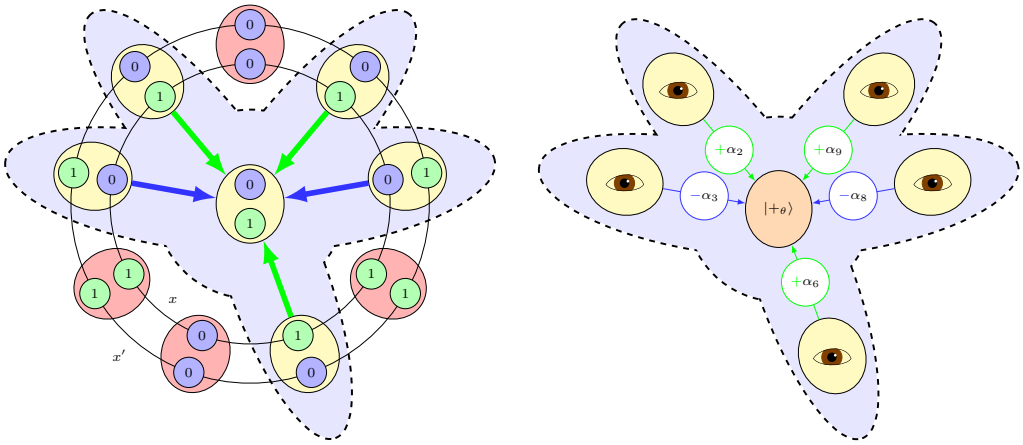
(where we no longer keep the qubits in order, but group them depending on their belonging to G or $\bar{G}$):

$$\left(\otimes_{i\in\bar{G}} |x_i\rangle \right) \otimes \left(\otimes_{j\in G} |x_j\rangle + \otimes_{j\in G} |x_j \oplus 1\rangle \right) \tag{5}$$

This can be rewritten as (up to trivial re-normalization):

$$\left(\otimes_{i\in\bar{G}} |x_i\rangle \right) \otimes \left(\prod_{j\in G} X^{x_j} \right) \left(|0\cdots 0\rangle_G + |1\cdots 1\rangle_G \right) \tag{6}$$

It is now evident that the state at the end of Stage 1 is a tensor product of isolated $|0\rangle$ and $|1\rangle$ states, and a GHZ state with random X's applied. You can find in Figure 1 an illustration of this state, before and after the Stage 2. Note that GHZ-states when viewed as graph states correspond to stars due to the corresponding graph as we can see here too.



(**a**) (**b**)

Figure 1. A simplified representation of the protocol. The red and yellow ellipses represent the qubits, the inner circle contains the bits of x and the outer circle contains the bits of x'. The central qubit is the last one, which is not measured and which will be the output qubit. (**a**) End of Stage 1: The yellow qubits are in a big Greenberger–Horne–Zeilinger (GHZ)-like state. The server does not know which qubits are in the GHZ state, and which qubits are not (in red). (**b**) End of Stage 2: Measuring any qubit in the GHZ state will rotate the last (output) qubit depending on the angle (and result) of the measurement.

The important thing to note is that the set G, that determines which qubits are in the GHZ state and which qubits are not, is not known to the server (apart from the fact that the position of the output qubit belongs to G since otherwise the protocol aborts). Moreover, this set denotes the positions where x and x' differ, which is given by the XOR of the two preimages $x \oplus x' := (x_1 \oplus x'_1, \cdots, x_n \oplus x'_n)$. As a result of second preimage resistance of the function, the server should not be able to invert and obtain $x \oplus x'$ apart with negligible probability (without access to the trapdoor t_k). This in itself does not guarantee that the server cannot learn any information about the XOR of the preimages, but we will see that the actual form of the state is such that being able to obtain information would lead to invert the full XOR and thus break the second preimage resistance.

Now, let us continue towards Stage 2. Measuring a qubit (other than the last one) in $\bar{G}$ has no effect on the last qubit (since it is disentangled). When the qubit index is in G, then measuring it at angle $\alpha_i \pi/4$ gives a phase to the output qubit of the form $(-(-1)^{x_i}\alpha_i + 4b_i)\pi/4$ as one can easily check. We remark that the $(-1)^{x_i}$-term arises because of the commutation of $X_i^{x_i}$ with the measurement angle, and the final $X_n^{x_n}$ gate

gives an overall $(-1)^{x_n}$ to the angle of deviation. Therefore, adding all the phases leads to the output state being:

$$|+_\theta\rangle;\ \theta = \frac{\pi}{4}(-1)^{x_n}\left(\sum_{i\in G\setminus\{n\}}\big(-\alpha_i(-1)^{x_i}+4b_i\big)\right) \bmod 8 \tag{7}$$

As a result that θ is defined modulo 2π and $-4 = 4 \bmod 8$, we can express the output angle in a more symmetrical way:

$$\theta = \frac{\pi}{4}(-1)^{x_n}\left(\sum_{i=1}^{n-1}(x_i - x'_i)\big(4b_i+\alpha_i\big)\right) \bmod 8 \tag{8}$$

Note that because the angles are defined modulo 2π, one can represent this angle as a 3-bits string $\tilde{B}$ (interpretable as an integer) such that $\theta := \tilde{B} \times \frac{\pi}{4}$ and eventually remove the $(-1)^{x_n}$ if needed by choosing the suitable convention in defining x and x'. □

A final remark is that in an honest run of this protocol, the measurement outcomes b_i and y are uniformly chosen from $\{0,1\}$ and $\mathrm{Im}(f_k)$ respectively. This justifies why in the honest-but-curious model we can view the protocol as sampling randomly the different α, y, b's.

5. Security of $\mathsf{HBC-QFactory}$

Here we will prove the security of $\mathsf{HBC-QFactory}$ (Algorithm 2) against honest-but-curious adversaries in the game-based security model.

Before proceeding further, it is worth stressing that this security level has three-fold importance. Firstly, the honest-but-curious model concerns any application of $\mathsf{CC-RSP}_\theta$ that involves a protocol where the adversaries are third parties that have access to the classical communication and nothing else. In this case, we can safely assume that the quantum part of the protocol is followed honestly and we only require to prove that the third parties learn nothing about the classical description of the state from the classical public communication. Second case of interest is scenarios where the "server" does not intend to sabotage/corrupt the computation but may be interested to learn (for free) extra information. In such a case, the protocol should be followed honestly, since any non-reversible deviation other than copying classical information could corrupt the computation. Finally, the honest-but-curious case, as in the classical setting, is a first step towards proving the full security against malicious adversaries.

5.1. Game-Based Security Definition

Definition 11. *In the following,* $\mathsf{HBC-QFactory}$ *is said to be secure if for all QPT adversaries* $\mathcal{A}$ *the following game is won with probability at most* $\frac{1}{2} + \mathsf{negl}(n)$*:*

Game G_{sec}

Challenger		**Adversary** $\mathcal{A}$
$c \leftarrow_\$ \{0,1\}$		
$(\tilde{k}, \tilde{t_k}) \leftarrow_\$ Gen_{\mathcal{F}}(1^n)$		
$\tilde{\alpha} \leftarrow_\$ \{0,1\}^{3(n-1)}$		
$\tilde{x} \leftarrow_\$ Dom(f_{\tilde{k}})$		
$\tilde{y} \leftarrow f_{\tilde{k}}(\tilde{x})$		
$\tilde{b} \leftarrow_\$ \{0,1\}^{n-1}$		
$\theta^{(0)} \leftarrow Inv(\tilde{t_k}, \tilde{y}, \tilde{b})$		
$\theta^{(1)} \leftarrow_\$ \{0,1\}^3 \cdot \frac{\pi}{4} - \{\theta^{(0)}\}$		
	$\xrightarrow{\tilde{k}, \tilde{\alpha}, \tilde{y}, \tilde{b}, \theta^{(c)}, \lvert +_{\theta^{(c)}} \rangle}$	
	$\xleftarrow{\tilde{c}}$	

Adversary $\mathcal{A}$ wins the game G_{sec} if and only if $c = \tilde{c}$.

Then what we want to show is that:

$$Pr[\mathcal{A} \text{ wins } G_{sec}] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{9}$$

In other words:

$$Pr[\mathcal{A}(\theta^{(c)}, \lvert +_{\theta^{(c)}} \rangle) = c] \leq \frac{1}{2} + \mathsf{negl}(n) \tag{10}$$

Remark 2. *Note that a stronger game could have been defined, where the* $\lvert +_\theta \rangle$ *that is sent to the adversary is always* $\lvert +_{\theta^{(0)}} \rangle$*, along with* $\theta^{(c)}$*. This would be closer to the actual view of an honest-but-curious adversary, but it is not possible to prove the security of this game, no matter how secure the underlying protocol is. Indeed, given* $\theta^{(c)}$ *and* $\lvert +_{\theta^{(0)}} \rangle$*, it is always possible to measure the* $\lvert +_{\theta^{(0)}} \rangle$ *in the basis* $\theta^{(c)}$ *and output* $\tilde{c} = 0$ *only if the measurement outcome was 0.*

The case $c = 0$ corresponds to the adversary receiving a transcript of the protocol $(\tilde{k}, \tilde{y}, \tilde{b})$ along with the real output of the protocol $(\theta^{(0)}, \lvert +_{\theta^{(0)}} \rangle)$ matching to this transcript. The case $c = 1$ corresponds to adversary receiving a transcript of the protocol along with a random output of the protocol $(\theta^{(1)}, \lvert +_{\theta^{(1)}} \rangle)$. The game ensures that an adversary cannot distinguish these two views. The security of G_{sec} tries to capture the following idea: If an adversary (server) runs honestly the protocol and keeps a record of the measurements he performs together with the transcripts he receives during the protocol, it must be hard for him to correlate this internal view with the output of the protocol $(\theta, \lvert +_\theta \rangle)$. This is why in the game we ask the adversary to distinguish whether he has access to either a correlated or an uncorrelated θ.

5.2. Game-Based Security of HBC − QFactory

Theorem 6. *For any QPT adversary* $\mathcal{A}$*, the game* G_{sec} *can be won with probability at most* $\geq \frac{1}{2} + \mathsf{negl}(n)$*.*

To prove that HBC − QFactory is secure according to Definition 11, we will rely on the following result, proven in the next section:

Theorem 7. *The function θ expressed as:*

$$\theta = \frac{\pi}{4}\left(\sum_{i=1}^{n-1}(x_i - x_i')(4b_i + \alpha_i)\right) \bmod 8 \tag{11}$$

as was defined in Algorithm 2, is a hard-core function with respect to f_k.
NB: here the collision resistance is not needed and is replaced by the weaker second preimage resistance property.

Proof of Theorem 6. The main idea would be to use a reduction to the hardcore property of the state description θ. We will assume that there exists an adversary $\mathcal{A}$ that can win G_{sec} with probability $\frac{1}{2} + p$ and we will construct an adversary $\mathcal{A}'$ that can break the hard-core function property of θ with probability $\frac{1}{8} + \frac{p}{4}$. This implies that if the game G_{sec} can be won with inverse polynomial probability, the same applies to the hard-core function property, and hence we reach a contradiction.

More specifically, let us assume that the adversary $\mathcal{A}$ can win with probability $\frac{1}{2} + p_0$ when $c = 0$ and with probability $\frac{1}{2} + p_1$ when $c = 1$. Then, we have:

$$Pr[\mathcal{A} \text{ wins } G_{sec}] = Pr[\tilde{c} = c] = Pr[\tilde{c} = 0 \,|\, c = 0] \cdot \frac{1}{2} + Pr[\tilde{c} = 1 \,|\, c = 1] \cdot \frac{1}{2}$$

So we have: $Pr[\tilde{c} = 0 \,|\, c = 0] + Pr[\tilde{c} = 1 \,|\, c = 1] = 1 + 2p$.
By denoting $Pr[\tilde{c} = 0 \,|\, c = 0] = \frac{1}{2} + p_0$ and $Pr[\tilde{c} = 1 \,|\, c = 1] = \frac{1}{2} + p_1$, we obtain: $p_0 + p_1 = 2p$.

Now we can define the following adversary $\mathcal{A}'$ that will attack the hardcore property of the state description using the adversary $\mathcal{A}$ for the game G_{sec}. Namely, $\mathcal{A}'$ will receive the tuple (y, b, k)—representing the messages that an adversary has during an honest run of $\mathsf{HBC} - \mathsf{QFactory}$ and will try to determine the underlying state description θ.

$\mathcal{A}'(y, b, k, \alpha)$

$\phi \leftarrow_\$ \{0,1\}^3 \cdot \frac{\pi}{4}$
Sends $(k, \alpha, y, b, \phi, |+_\phi\rangle)$ to $\mathcal{A}$
$\tilde{c} \leftarrow \mathcal{A}(k, \alpha, y, b, \phi, |+_\phi\rangle)$
if $(\tilde{c} == 0)$ **then**
  $\tilde{\theta} \leftarrow \phi$
elseif $(\tilde{c} == 1)$ **then**
  $\tilde{\theta} \leftarrow_\$ \{0,1\}^3 \cdot \frac{\pi}{4} - \{\phi\}$
  return $\tilde{\theta}$

Now we want to determine the probability that the output of the adversary $\mathcal{A}$ is equal to the "true" $\theta(y, b, k, \alpha)$ that is obtained by $Inv(t_k, y, b, \alpha)$.

We will consider separately two cases, namely: (i) $\phi = \theta(y, b, k, \alpha)$ and (ii) $\phi \neq \theta(y, b, k, \alpha)$. Since ϕ is chosen randomly from the eight possible angles, it is clear that case (i) occurs with probability $1/8$ and case (ii) occurs with probability $7/8$.

(i) $\mathcal{A}$ receives the "true" state ($c = 0$), so to win the game he needs to return $\tilde{c} = 0$. By definition, this happens with $\frac{1}{2} + p_0$, and in this case $\mathcal{A}'$ also wins (since he outputs the correct state). The overall probability that all this happens, i.e., that $\mathcal{A}'$ succeeds in this case, is $\frac{1}{8} \cdot \left(\frac{1}{2} + p_0\right)$.

(ii) $\mathcal{A}$ receives one of the "false" states ($c = 1$), and thus to win G_{sec} he needs to return $\tilde{c} = 1$. By definition this happens with probability $\frac{1}{2} + p_1$. Now, in this case, $\mathcal{A}'$ has essentially ruled-out one of the eight possible states. His random guess, after ruling-

out one state, succeeds with probability $\frac{1}{7}$. Combining all this together we see that $\mathcal{A}'$ succeeds with probability $\frac{7}{8} \cdot \left(\frac{1}{2} + p_1\right) \cdot \frac{1}{7}$.

More explicitly, the probability that $\mathcal{A}'$ breaks the hardcore property of θ is (where we denoted the "true" $\theta(y, b, k, \alpha)$ by simply θ):

$$\begin{aligned} &Pr[\mathcal{A}'(y,b,k,\alpha) = \theta] = Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi = \theta] \cdot Pr[\phi = \theta] + \\ &+ Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi \neq \theta] \cdot Pr[\phi \neq \theta] \\ &= \frac{1}{8} \cdot Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi = \theta] + \frac{7}{8} \cdot Pr[\mathcal{A}'(y,b,k,\alpha) = \theta \,|\, \phi \neq \theta] \\ &= \frac{1}{8} Pr[\tilde{c} = 0 \,|\, c = 0] + \frac{7}{8} Pr[\tilde{c} = 1 \text{ and random } \tilde{\theta} = \theta | c = 1] \\ &= \frac{1}{8}\left(\frac{1}{2} + p_0\right) + \frac{7}{8} \cdot \left(\frac{1}{2} + p_1\right) \cdot \frac{1}{7} = \frac{1}{8} + \frac{p_0 + p_1}{8} = \frac{1}{8} + \frac{p}{4} \end{aligned} \tag{12}$$

We showed that Adversary $\mathcal{A}'$ succeeds with probability $\frac{1}{8} + \frac{p}{4}$ in guessing $\theta(y, b, k, \alpha)$, where p is the advantage $\mathcal{A}$ has in G_{sec}. However, we will prove in Theorem 7, that $\mathcal{A}$ can't win with probability better than $1/8 + \mathsf{negl}(n)$. Contradiction. □

5.3. Hardcore Function θ

Proof Sketch of Theorem 7. In Algorithm 2, the adversary (server) can only use the classical information that he possesses (k, y, α, b) in order to try and guess with some probability the value of θ in the case that there is no abort. Since the adversary follows the honest protocol, the choices of y, b are truly random (and not determined by the adversary as he could in the malicious case). **Outline of proof sketch:** We first express the classical description of the state into expressions for each of the corresponding three bits. The aim is to prove that it is impossible to distinguish the sequence of these three bits from three random bits with non-negligible probability. To show this we follow five steps. In **Step 1**, we express each of the bits as a sum mod two, of an inner product (of the form present in GL theorem) and some other terms. In **Step 2**, we show that guessing the sum modulo two of the two preimages breaks the second preimage resistance of the function and thus is impossible. We assume that the adversary can achieve some inverse polynomial advantage in guessing certain predicates and in the remaining steps we show that in this case he can obtain a polynomial inversion algorithm for the one-way function f_k, and thus reach the contradiction. In **Step 3**, we use the Vazirani–Vazirani Theorem 3 to reduce the proof of hard-core function to a number of single hard-core bits (predicates). In **Step 4**, we use a result that allows us to fix all but one variable in each expression, with an extra cost that is an inverse polynomial probability and therefore the (fixed variables) guessing algorithm still needs to have negligible success probability. Finally, in **Step 5**, we reduce all the predicates in a form of a known hard-core predicate XOR with a function that involves variables not included in that predicate. Using the previous step, it reduces to guessing the XOR of a hard-core predicate with a constant, which is bounded by the probability of guessing the (known to be hard-core) predicate.

Here we give the sketch described above, while the full proof can be found in the Appendix B. Let us start by defining:

$$\widetilde{B} = \widetilde{B}_1 \widetilde{B}_2 \widetilde{B}_3 = \left(\sum_{i=1}^{n-1} (x_i - x'_i)(4b_i + \alpha_i) \right) \bmod 8 \tag{13}$$

where $\widetilde{B}_i$ are single bits. Moreover, we treat x, x' as vectors in $\{0,1\}^n$; we define $\alpha^{(j)} = (\alpha_1^{(j)}, \cdots, \alpha_{n-1}^{(j)})$ the vector that involves the $j \in \{1,2,3\}$ bit of each of three-bit strings α, and we define $\tilde{x} := x \oplus x'$. We define z as a vector in $\{-1,0,1\}^n$ defined as the element-wise differences of the bits of x and x', i.e., $z_i = x_i - x'_i$. Finally, as in GL theorem, we use the notation for the inner product $\langle a, b \rangle = \sum_{i=1}^{n-1} a_i b_i$.

We will prove that any QPT adversary $\mathcal{A}$ having all the classical information that server has (y, α, b), can guess $\widetilde{B}$ with at most negligible probability:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ \alpha\leftarrow\{0,1\}^{3n} \\ b\leftarrow\{0,1\}^n}} [\mathcal{A}(f(x), \alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, b) = \widetilde{B_1}\widetilde{B_2}\widetilde{B_3}] \leq \frac{1}{8} + \mathsf{negl}(n) \tag{14}$$

where for simplicity we denote the function f instead of f_k. This means that the adversary $\mathcal{A}$ cannot distinguish $\widetilde{B}$ from a random three-bit string with non-negligible probability.

Step 1: We decompose Equation (13) into three separate bits, and use the variable $\tilde{x}, z$ defined above.

$$\begin{aligned} \widetilde{B}_3 &= \langle \tilde{x}, \alpha^{(3)} \rangle \bmod 2 \\ \widetilde{B}_2 &= \langle \tilde{x}, \alpha^{(2)} \rangle \bmod 2 \oplus h_2(z, \alpha^{(3)}) \\ \widetilde{B}_1 &= \langle \tilde{x}, \alpha^{(1)} \rangle \bmod 2 \oplus h_1(z, \alpha^{(3)}, \alpha^{(2)}, b) \end{aligned} \tag{15}$$

where the derivation and exact expressions for the functions h_1, h_2 are given in Appendix B. We notice from Equation (15) that each bit includes a term of the form $\langle \tilde{x}, \alpha^{(i)} \rangle \bmod 2$ which on its own is a hard-core predicate following the GL theorem.

Step 2: By the second preimage resistance we have:

$$\begin{aligned} &\Pr_{x\leftarrow\{0,1\}^n} [\mathcal{A}(1^n, x) = x' \text{ such that } f(x) = f(x') \text{ and } x \neq x'] \leq \mathsf{negl}(n) \Rightarrow \\ &\Pr_{x\leftarrow\{0,1\}^n} [\mathcal{A}(1^n, x) = x' \oplus x = \tilde{x}] \leq \mathsf{negl}(n) \end{aligned} \tag{16}$$

For each bit $j \in \{1, 2, 3\}$, separately we assume that the adversary can achieve an advantage in guessing $\widetilde{B}_j$ which is $\frac{1}{2} + \varepsilon_j(n)$. Then, similarly to GL theorem, we prove that if this $\varepsilon_j(n)$ is inverse polynomial, this leads to contradiction with Equation (16) since one can obtain an inverse-polynomial inversion algorithm for the one-way function f.

Step 3: While each bit includes terms that on its own it would make it hard-core predicate (as stated in Step 1), if we XOR the overall bit with other bits it could destroy this property. To proceed with the proof that $\widetilde{B}$ is hard-core function, we use the Vazirani–Vazirani theorem which states that it suffices to show that individual bits as well as combinations of XOR's of individual bits are all hard-core predicates. In this way, one evades the need to show explicitly that the guesses for different bits are not correlated. To proceed with the proof, we use a trick that "disentangles" the different variables.

Step 4: We would like to be able to fix one variable and vary only the remaining, while at the same time maintain some bound on the guessing probability.

The advantage $\varepsilon_j(n)$ that we assume the adversary has for guessing one bit (or a XOR) is calculated "on average" over all the random choices of $(\tilde{x}, \alpha^{(i)}, b)$. Using Lemma 1 we can fix one-by-one all but one variable (applying the lemma iteratively, see Appendix B). With suitable choices, the cardinality of the set of values that satisfies all these conditions is $O(2^n \varepsilon_j(n))$ for each iteration. Unless $\varepsilon_j(n)$ is negligible, this size is an inverse polynomial fraction of all values. This suffices to reach the contradiction. The actual inversion probability that we will obtain is simply a product of the extra cost of fixing the variables with the standard GL inversion probability. This extra cost is exactly the ratio between the cardinality of the *Good* sets (defined below) and the set of all values.

Lemma 1. *Let* $\Pr_{(v_1,\cdots,v_k)\leftarrow\{0,1\}^n\times\cdots\times\{0,1\}^n}[Guessing] \geq p+\varepsilon(n)$*, then for any variable* v_i*, there exists a set* $Good_{v_i} \subseteq \{0,1\}^n$ *of size at least* $\frac{\varepsilon(n)}{2}2^n$*, such that for all* $v_i \in Good_{v_i}$*, we have:*

$$\Pr_{(v_1,\cdots,\not{v_i},\cdots,v_k)\leftarrow\{0,1\}^n\times\cdots\times\{0,1\}^n}[Guessing] \geq p+\frac{\varepsilon(n)}{2}$$

where the latter probability is taken over all variables except v_i*.*

Step 5: If the expression we wish to guess involves XOR of terms that depend on different variables, then by using Step 4 we can fix the variables of all but one term. Then we note that trying to guess a bit (that depends on some variable and has expectation value close to 1/2) is at least as hard as trying to guess the XOR of that bit with a constant. For example, if the bit we want to guess is $\langle\widetilde{x},r_1\rangle \bmod 2 \oplus h(z,r_2,r_3)]$ and we have a bound on the guessing probability where only r_1 is varied, then we have

$$\Pr_{r_1\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r_1,r_2,r_3) = \langle\widetilde{x},r_1\rangle \bmod 2 \oplus h(z,r_2,r_3)] \leq \Pr_{r_1\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r_1,r_2,r_3) = \langle\widetilde{x},r_1\rangle \bmod 2]$$

We note that all bits of $\widetilde{B}$ and their XOR's can be brought in this form. Then using this, we can now prove security, as the r.h.s. is exactly in the form where the GL theorem provides an inversion algorithm for the one-way function f. For details, see Appendix B. □

6. Function Constructions

For our Algorithm 2, we need a trapdoor one-way function that is also quantum-safe, two-regular and second preimage resistant (or the stronger collision resistance property). These properties may appear to be too strong to achieve, however, we give here methods to construct functions that achieve these properties starting from trapdoor one-way functions that have fewer (more realistic) conditions, and we specifically give one example that achieves all the desired properties. In particular, we give:

- A general construction given either (i) an injective, homomorphic (with respect to any operation and in particular it is only required to be homomorphic once for this operation) trapdoor one-way function or (ii) a bijective trapdoor one-way function, to obtain a two-regular, second preimage resistant , trapdoor one-way function. In both cases the quantum-safe property is maintained (if the initial function has this property, so does the constructed function). We note that for (i) we prove the stronger collision resistant property.
- (Taken from [44]) A method of how to realise injective quantum-safe trapdoor functions derived from the LWE problem, that has certain homomorphic property.
- A way to use the first construction with the trapdoor from [44] that requires a number of modifications, including relaxation of the notion of two-regularity. The resulting function satisfies all the desired properties if a choice of parameters satisfying certain constraints can be found.
- A specific choice of these parameters satisfying all constraints, that leads to a concrete function with all the desired properties.

6.1. Obtaining Two-Regular, Collision Resistant/Second Preimage Resistant, Trapdoor One-Way Functions

Here we give two constructions. The first uses as a starting point an injective, homomorphic trapdoor function while the second a bijective trapdoor function. While we give both constructions, we focus on the first construction since (i) we can prove the stronger collision-resistance property and (ii) (to our knowledge) there is no known bijective trapdoor function that is believed to be quantum-safe.

Theorem 8. *If $\mathcal{G}$ is a family of injective, homomorphic, trapdoor one-way functions, then there exists a family $\mathcal{F}$ of two-regular, collision resistant, trapdoor one-way functions. Moreover, the family $\mathcal{F}$ is quantum-safe if and only if the family $\mathcal{G}$ is quantum-safe.*

From now on, we consider that any function $g_k \in \mathcal{G}$ has domain D and range R and let $+_D$ be the closed operation on D and $+_R$ be the closed operation on R such that g_k is the morphism between D and R with respect to these 2 operations:

$$g_k(a) +_R g_k(b) = g_k(a +_D b) \quad \forall a, b \in D$$

We also denote the operation $-_D$ on D, the inverse operation of $+_D$, specifically: $a +_D b^{-1} = a -_D b \quad \forall a, b \in D$ and 0_D be the identity element for $+_D$.

Then, the family $\mathcal{F}$ is described by the following PPT algorithms:

$\texttt{FromInj.Gen}_{\mathcal{F}}(1^n)$

1: $(k, t_k) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$ // k is an index of a function from $\mathcal{G}$ and t_k is its associated trapdoor
2: $x_0 \leftarrow_{\$} D \setminus \{0_D\}$ // $x_0 \neq 0_D$ to ensure that the 2 preimages mapped to the same output are distinct
3: $k' := (k, g_k(x_0))$ // the description of the new function
4: $t'_k := (t_k, x_0)$ // the trapdoor associated with the function $f_{k'}$
5: **return** k', t'_k

The evaluation procedure receives as input an index k' of a function from $\mathcal{F}$ and an element $\bar{x}$ from the function's domain ($\bar{x} \in D \times \{0,1\}$):

$\texttt{FromInj.Eval}_{\mathcal{F}}(k', \bar{x})$

return $f_{k'}(\bar{x})$

where every function from $\mathcal{F}$ is defined as:

$$f_{k'} : D \times \{0,1\} \to R$$
$$f_{k'}(x, c) = \begin{cases} g_k(x), & \text{if } c = 0 \\ g_k(x) +_R g_k(x_0) = g_k(x +_D x_0), & \text{if } c = 1 \end{cases}$$

The last equality follows since each function g_k from $\mathcal{G}$ is homomorphic.

$\texttt{FromInj.Inv}_{\mathcal{F}}(k', y, t'_k)$

1: // y is an element from the image of $f_{k'}$, $k' = (k, g_k(x_0))$, $t'_k = (t_k, x_0)$
2: $x_1 := Inv_{\mathcal{G}}(k, y, t_k)$
3: $x_2 := x_1 -_D x_0$
4: **return** $(x_1, 0)$ *and* $(x_2, 1)$ // the unique 2 preimages corresponding to
5: // an element from the image of $f_{k'}$

Proof. To prove Theorem 8 we give below five lemmata showing that, the family $\mathcal{F}$ of functions defined above, satisfies the following properties: (i) two-regular, (ii) trapdoor, (iii) one-way, (iv) collision resistant and (v) quantum-safe. □

Lemma 2 (two-regular)**.** *If $\mathcal{G}$ is a family of injective, homomorphic functions, then $\mathcal{F}$ is a family of two-regular functions.*

Proof. For every $y \in \operatorname{Im} f_{k'} \subseteq R$, where $k' = (k, g_k(x_0))$:

1. Since $\operatorname{Im} f_{k'} = \operatorname{Im} g_k$ and g_k is injective, there exists a unique $x := g_k^{-1}(y)$ such that $f_{k'}(x, 0) = g_k(x) = y$.

2. Assume x' such that $f_{k'}(x',1) = y$. By definition $f_{k'}(x',1) = g_k(x' +_D x_0) = y$, but g_k is injective and $g_k(x) = y$ by assumption, therefore there exists a unique $x' = x -_D x_0$ such that $f_{k'}(x',1) = y$

Therefore, we conclude that:

$$\forall\, y \in \operatorname{Im} f_{k'} :\ f_{k'}^{-1}(y) := \{(g_k^{-1}(y), 0), (g_k^{-1}(y) -_D x_0, 1)\} \tag{17}$$

□

Lemma 3 (trapdoor)**.** *If $\mathcal{G}$ is a family of injective, homomorphic, trapdoor functions, then $\mathcal{F}$ is a family of trapdoor functions.*

Proof. Let $y \in \operatorname{Im} f_{k'} \subseteq R$. We construct the following inversion algorithm:

$Inv_{\mathcal{F}}(k', y, t'_k)$

1 : // $t'_k = (t_k, x_0), k' = (k, g_k(x_0))$
2 : $x := Inv_{\mathcal{G}}(k, y, t_k)$
3 : **return** $(x, 0)$ and $(x -_D x_0, 1)$

□

Lemma 4 (one-way)**.** *If $\mathcal{G}$ is a family of injective, homomorphic, one-way functions, then $\mathcal{F}$ is a family of one-way functions.*

Proof. We prove it by contradiction. We assume that there exists a QPT adversary $\mathcal{A}$ that can invert a function in $\mathcal{F}$ with non-negligible probability P (i.e., given $y \in \operatorname{Im} f_{k'}$ to return a correct preimage of the form (x', b) with probability P). We then construct a QPT adversary $\mathcal{A}'$ that inverts a function in $\mathcal{G}$ with the same non-negligible probability P reaching a contradiction, since $\mathcal{G}$ is one-way by assumption.

From Equation (17) of Lemma 2, we know the two preimages of y are: (i) $(g_k^{-1}(y), 0)$ and (ii) $(g_k^{-1}(y) -_D x_0, 1)$. We see that information on $g_k^{-1}(y)$ is obtain in both cases, i.e., obtaining any of these two preimages, is sufficient to recover $g_k^{-1}(y)$ if x_0 is known. We now construct an adversary $\mathcal{A}'$ that for any function $g_k : D \to R$, inverts any output $y = g_k(x)$ with the same probability P that $\mathcal{A}$ succeeds.

$\mathcal{A}'(k, y)$

1 : $x_0 \leftarrow_\$ D \setminus \{0_D\}$ // $\mathcal{A}'$ knows x_0, but is not given to $\mathcal{A}$
2 : $k' := (k, g_k(x_0))$
3 : $(x', b) \leftarrow \mathcal{A}(k', y)$
4 : **if** $((b == 0) \wedge (g_k(x') == y)$ **then**
5 : // equivalent to $\mathcal{A}$ succeeded in returning the first preimage
6 : **return** x'
7 : **elseif** $((b == 1) \wedge (g_k(x' +_D x_0)) == y)$ **then**
8 : // $\mathcal{A}$ succeeded in returning the second preimage
9 : **return** $x' +_D x_0$ // $\mathcal{A}'$ uses x_0 known from step 1
10 : **else** // $\mathcal{A}$ failed in giving any of the preimages (happens with probability $1 - P$)
11 : **return** 0

□

Lemma 5 (collision-resistance)**.** *If $\mathcal{G}$ is a family of injective, homomorphic, one-way functions, then any function $f \in \mathcal{F}$ is collision resistant.*

Proof. Assume there exists a QPT adversary $\mathcal{A}$ that given $k' = (k, g_k(x_0))$ can find a collision $(y, (x_1, b_1), (x_2, b_2))$ where $f_{k'}(x_1, b_1) = f_{k'}(x_2, b_2) = y$ with non-negligible probability P. From Equation (17), we know that the two preimages are of the form $(x, 0), (x \triangle x_0, 1)$ where $g_k(x) = y$. It follows that when $\mathcal{A}$ is successful, by comparing the first arguments of the two preimages, can recover x_0.

We now construct a QPT adversary $\mathcal{A}'$ that inverts the function g_k with the same probability P, reaching a contradiction:

$\mathcal{A}'(k, g_k(x))$

1 : $k' := (k, g_k(x))$
2 : $(y, (x_1, b_1), (x_2, b_2)) \wedge x_1 \neq x_2 \leftarrow \mathcal{A}(k')$ // where y is an element from the image of $f_{k'}$
3 : **if** $f(x_1, b_1) == f(x_2, b_2) == y$
4 : **return** $x := x_1 \ -_D \ x_2$
5 : **else** // $\mathcal{A}$ failed to find collision of $f_{k'}$; happens with probability $(1 - P)$
6 : **return** 0

□

Lemma 6 (quantum-safe)**.** *If $\mathcal{G}$ is a family of quantum-safe trapdoor functions, with properties as above, then $\mathcal{F}$ is also a family of quantum-safe trapdoor functions.*

Proof. The properties that require to be quantum-safe is one-wayness and collision resistance. Both these properties of $\mathcal{F}$ that we derived above were proved using reduction to the hardness (one-wayness) of $\mathcal{G}$. Therefore, if $\mathcal{G}$ is quantum-safe, its one-wayness is also quantum-safe and thus both properties of $\mathcal{F}$ are also quantum-safe. □

Theorem 9. *If $\mathcal{G}$ is a family of bijective, trapdoor one-way functions, then there exists a family $\mathcal{F}$ of two-regular, second preimage resistant, trapdoor one-way functions. Moreover, the family $\mathcal{F}$ is quantum-safe if and only if the family $\mathcal{G}$ is quantum-safe.*

The family $\mathcal{F}$ is described by the following PPT algorithms, where each function $g_k \in \mathcal{G}$ has domain D and range R:

$\texttt{FromBij.Gen}_{\mathcal{F}}(1^n)$

1 : $(k_1, t_{k_1}) \leftarrow\!\!\$\ Gen_{\mathcal{G}}(1^n)$
2 : $(k_2, t_{k_2}) \leftarrow\!\!\$\ Gen_{\mathcal{G}}(1^n)$
3 : $k' := (k_1, k_2)$
4 : $t'_k := (t_{k_1}, t_{k_2})$
5 : **return** k', t'_k

$\texttt{FromBij.Eval}_{\mathcal{F}}(k', \bar{x})$

return $f_{k'}(\bar{x})$

where every function from $\mathcal{F}$ is defined as:

$$f_{k'} : D \times \{0,1\} \to R$$
$$f_{k'}(x, c) = \begin{cases} g_{k_1}(x), & \text{if } c = 0 \\ g_{k_2}(x), & \text{if } c = 1 \end{cases}$$

$\texttt{FromBij.Inv}_{\mathcal{F}}(k', y, t'_k)$

1 : // y is an element from the image of $f_{k'}$, $k' = (k_1, k_2)$, $t'_k = (t_{k_1}, t_{k_2})$
2 : $x_1 := Inv_{\mathcal{G}}(k_1, y, t_{k_1})$
3 : $x_2 := Inv_{\mathcal{G}}(k_2, y, t_{k_2})$
4 : **return** $(x_1, 0)$ *and* $(x_2, 1)$ // the unique 2 preimages corresponding to
5 : // an element from the image of $f_{k'}$

The proof of Theorem 9, using the family of function defined above, follows same steps as of Theorem 8 and is given in the Appendix C.

6.2. Injective, Homomorphic Quantum-Safe Trapdoor One-Way Function from LWE

We outline the Micciancio and Peikert [44] construction of injective trapdoor one-way functions, naturally derived from the Learning-With-Errors problem. At the end we comment on the homomorphic property of the function, since this is crucial in order to use this function as the basis to obtain our desired two-regular, collision resistant trapdoor one-way functions.

The algorithm below generates the index of an injective function and its corresponding trapdoor. The matrix G used in this procedure, is a fixed matrix (whose exact form can be seen in [44]) for which the function from the family $\mathcal{G}$ with index G can be efficiently inverted without any trapdoor.

$\texttt{LWE.Gen}_{\mathcal{G}}(1^n)$

1 : $A' \leftarrow_{\$} \mathbb{Z}_q^{n \times \bar{m}}$
2 : $R \leftarrow_{\$} \mathcal{D}_{\alpha q}^{\bar{m} \times kn}$ // element-wise gaussian distribution with mean 0, standard deviation αq on $\bar{m} \times kn$ matrices
3 : $A := (A', G - A'R)$ // concatenation of matrices A' and $G - A'R$, representing the index of the function
4 : **return** (A, R) // A—public function index, R—trapdoor

where the parameters k, α, q and $\bar{m}$ are defined in Theorem 11.

The actual description of the injective trapdoor function is given in the evaluation algorithm below, where each function from $\mathcal{G}$ is defined on: $g_K : \mathbb{Z}_q^n \times L^m \to \mathbb{Z}_q^m$, and L is the domain of the errors in the LWE problem (the set of integers bounded in absolute value by the parameter μ):

$\texttt{LWE.Eval}_{\mathcal{G}}(K, (s, e))$

1 : $y := g_K(s, e) = s^t K + e^t$
2 : **return** y

The inversion algorithm returns the unique preimage (s, e) corresponding to $b^t \in \mathrm{Im}(g_K)$. The algorithm uses as a subroutine the efficient algorithm Inv_G for inverting the function g_G, with G the fixed matrix mentioned before.

$\texttt{LWE.Inv}_{\mathcal{G}}(K, t_K, b^t)$

1 : $b'^t := b^t \begin{bmatrix} R \\ I \end{bmatrix}$
2 : $(s', e') := Inv_G(b')$
3 : $s := s'$
4 : $e := b - K^t s$
5 : **return** s, e

We examine now whether the functions g_K are homomorphic with respect to some operation.

We first define the domain and the range as $D := \mathbb{Z}_q^n \times L^m$ and $R := \mathbb{Z}_q^m$. Then, given $a = (s_1, e_1) \in \mathbb{Z}_q^n \times L^m$ and $b = (s_2, e_2) \in \mathbb{Z}_q^n \times L^m$, the operation $+_D$ is defined as:

$$(s_1, e_1) +_D (s_2, e_2) = (s_1 + s_2 \bmod q, e_1 + e_2)$$

Given $y_1 = g_K(a) \in \mathbb{Z}_q^m$ and $y_2 = g_K(b) \in \mathbb{Z}_q^m$, the operation $+_R$ is defined as:

$$y_1 +_R y_2 = y_1 + y_2 \bmod q$$

Then, we can easily verify that:

$$g_K(s_1, e_1) + g_K(s_2, e_2) \bmod q = {s_1}^t K + {e_1}^t + {s_2}^t K + {e_2}^t \bmod q = (s_1 + s_2 \bmod q)^t K + (e_1 + e_2)^t = g_K((s_1 + s_2) \bmod q, e_1 + e_2)$$

However, the sum of two error terms, each being bounded by μ, may not be bounded by μ. This means that the function is not (properly) homomorphic. Instead, what we conclude is that as long as the vector $e_1 + e_2$ lies inside the domain of g_K, then g_K is homomorphic. To address this issue, we will need to define a weaker notion of 2-regularity, and a (slight) modification of the `FromInj` construction to provide a desired function starting from the trapdoor function of [44].

6.3. A Suitable δ-2 Regular Trapdoor Function

Using the injective trapdoor function of Micciancio and Peikert [44] and the construction defined in the proof of Theorem 8, we derive a family $\mathcal{F}$ of collision resistant trapdoor one-way function, but with a weaker notion of 2-regularity, called δ-2 regularity:

Definition 12 (δ-2 regular)**.** *A family of functions $(f_k)_{k \leftarrow Gen_{\mathcal{F}}}$ is said to be δ-2 regular, with $\delta \in [0, 1]$ if:*

$$\Pr_{k \leftarrow Gen_{\mathcal{F}}, y \in \mathrm{Im}(f_k)} [\, |f_k^{-1}(y)| = 2 \,] \geq \delta$$

Given this definition, we should note here that in Algorithm 2 we need to modify the abort case to include the possibility that the image y obtained from the measurement does not have two preimages (something that happens with at most probability $(1 - \delta)$).

Theorem 10 (Existence of a δ-2 regular trapdoor function family)**.** *There exists a family of functions that are δ-2 regular (with δ at least as big as a fixed constant), trapdoor, one-way, collision resistant and quantum-safe, assuming that there is no quantum algorithm that can efficiently solve* SIVP_γ *for* $\gamma = \mathsf{poly}(n)$.

Proof. To prove this theorem, we define a function similar to the one in the `FromInj` construction, where the starting point is the function defined in [44]. Crucial for the security is a choice of parameters that satisfy a number of conditions given by Theorem 11 and proven in Appendix D. The proof is then completed by providing a choice of parameters given in Lemma 7 that satisfies all conditions as it is shown in Appendix E. □

Definition 13. *For a given set of parameter $\mathcal{P}$ chosen as in Theorem 11, we define the following functions, that are similar to the construction* `FromInj`*, with the difference that the key generation requires the trapdoor error to be sampled from a smaller subdomain ($\alpha' < \alpha$):*

$\texttt{REG2.Gen}_{\mathcal{P}}(1^n)$

1: $(A, R) \leftarrow \texttt{LWE.Gen}_{\mathcal{G}}(1^n)$
2: $s_0 \leftarrow \mathbb{Z}_q^{n,1}$
3: $e_0 \leftarrow \mathcal{D}_{\alpha' q}^{m,1} / \alpha' < \alpha$
4: $b_0 := \texttt{LWE.Eval}(A, (s_0, e_0))$
5: $k := (A, b_0)$
6: $t_k := (R, (s_0, e_0))$
7: **return** (k, t_k)

$\texttt{REG2.Eval}_{\mathcal{P}}((A, b_0), (s, e, c))$

1: / *s is a random element in* $\mathbb{Z}_q^{n,1}$, $c \in \{0, 1\}$
2: / *e is sampled uniformly and such that*
3: / *each component is smaller than* μ
4: **return** $\texttt{LWE.Eval}(A, (s, e)) + c \cdot b_0$

$\texttt{REG2.Inv}_{\mathcal{P}}((A, R, (s_0, e_0)), b)$

1: $(s_1, e_1) := \texttt{LWE.Inv}(R, b)$
2: **if** $||e_1 - e_0||_\infty \leq \mu$ **then return** $\perp$
3: **return** $((s_1, e_1, 0), (s_1 - s_0 \bmod q, e_1 - e_0, 1))$

Note, that the pairs (s, e) and (s_0, e_0) correspond to x and x_0 of the $\texttt{FromInj}$ construction of Section 6.1. The idea behind this construction is that the noise of the trapdoor, e_0, is sampled from a set which is small compared to the noise of the input function. That way, when we will add the trapdoor (s_0, e_0) together with an input (s, e), the total noise will still be small enough to lie in the set of possible input noise with good probability, mimicking the homomorphic property needed in Theorem 8. Note that the parameters need to be carefully chosen, and a trade-off between probability of success and security exists.

We first introduce the following notation: for all $n, q, \mu \in \mathbb{Z}, \mu' \in \mathbb{R}$, let us define:

- $k := \lceil log(q) \rceil$,
- $\bar{m} = 2n$,
- $\omega = nk$,
- $m := \bar{m} + \omega = 2n + nk$,
- $\alpha' = \frac{\mu'}{\sqrt{mq}}$,
- $\alpha = m\alpha'$,
- C the constant in Lemma 2.9 of [44] which is approximately $\frac{1}{\sqrt{2\pi}}$,
- $B = 2$ if q is a power of 2, and $B = \sqrt{5}$ otherwise.

Theorem 11 (Requirements on the parameters)**.** *If for all security parameters n (dimension of the lattice), there exist q (the modulus of* LWE*) and μ (the maximum amplitude of the components of the errors) such that:*

1. *m is such that $n = o(m)$ (required for the injectivity of the function (see e.g., [50])),*
2. *$0 < \alpha < 1$,*
3. *$\mu' = O(\mu/m)$ (required to have non-negligible probability to have two preimages),*
4. *$\alpha' q \geq 2\sqrt{n}$ (required for the* LWE *to* SIVP *reduction),*
5. *$\frac{n}{\alpha'}$ is* poly(n) *(representing, up to a constant factor, the approximation factor γ in the* SIVP_γ *problem)—for the standard hardness of the* SIVP *problem.*
6. $$\sqrt{m}\mu < \underbrace{\frac{q}{2B\sqrt{\left(C \cdot (\alpha \cdot q) \cdot (\sqrt{2n} + \sqrt{kn} + \sqrt{n})\right)^2 + 1}}}_{r_{max}} - \mu'\sqrt{m}$$

   (required for the correctness of the inversion algorithm-r_{max} represents the maximum length of an error vector that one can correct using the [44] function, and the last term is needed in the proof of collision resistance to ensure injectivity even when we add the trapdoor noise, as illustrated in Figure 2. We remark that we chose to use the computational definition of [44], but this theorem can be easily extended to other definitions of the same paper, or even to other construction of trapdoor short basis).

Then, the family of functions of Definition 13 is δ-2 regular (with δ at least as big as a fixed constant), trapdoor, one-way and collision resistant (all these properties hold even against a quantum attacker), assuming that there is no quantum algorithm that can efficiently solve $SIVP_\gamma$ for $\gamma = \mathsf{poly}(n)$.

Proof. The proof follows by showing that the function with these constraints on the parameters is: (i) δ-2 regular, (ii) collision resistant, (iii) one-way and (iv) trapdoor. In Appendix D, we give and prove one lemma for each of those properties. For an intuition of the choice of parameters see also Figure 2. □

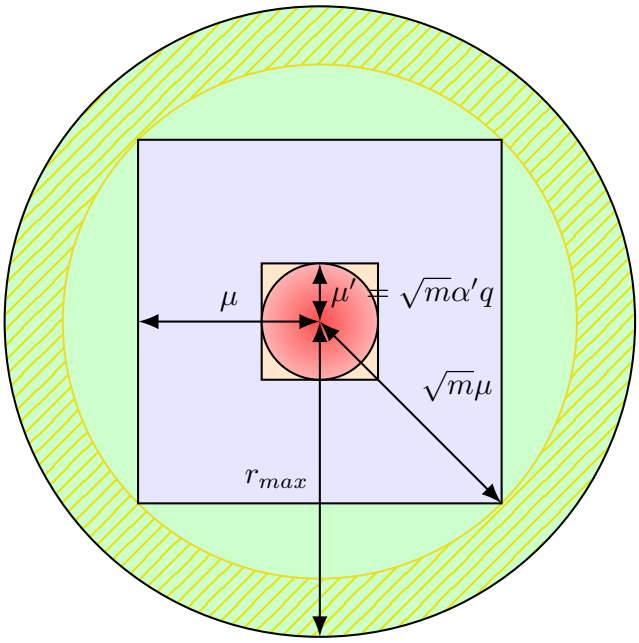


Figure 2. The red circle represents the domain of the error term from the trapdoor information, which is being sampled from a Gaussian distribution. The orange square is an approximation of this domain, which must satisfy that its length is much smaller (by a factor of at least m—the dimension of the error) than the length of the blue square, used for the actual sampling from the domain of the error terms, for which it is known that the trapdoor function is invertible, domain represented by the green circle (including the dashed part). The dashed part, representing the distance between the maximum domain for which the function is invertible (green circle) and the actual domain used for sampling (blue square) is needed to ensure that if there is a collision (x_1, x_2), then $x_1 = x_2 \pm x_0$.

6.4. Parameter Choices

Lemma 7 (Existence of parameters)**.** *The following set of parameters fulfils Theorem 11.*

$$
\begin{aligned}
n &= \lambda \\
k &= 5\lceil log(n) \rceil + 21 \\
q &= 2^k \\
\bar{m} &= 2n \\
\omega &= nk \\
m &= \bar{m} + \omega \\
\mu &= \left\lceil 2mn\sqrt{2+k} \right\rceil \\
\mu' &= \mu / m \\
B &= 2
\end{aligned}
$$

and α, α', C are defined like in Theorem 11.

The proof is given in Appendix E. As a final remark, we stress that other choices of the parameters are possible (considering the trade-off between security and probability of success) and we have not attempted to find an optimal set.

7. Implementation of HBC − QFactory on IBM Quantum Cloud

Finally, we provide the first proof-of-principle demonstration of a classical remote state preparation protocol. Very importantly, this implementation is not secure against a quantum adversary. The reason is that, as our 2-regular trapdoor function needs to be hard to invert this requires the size of the function to be sufficiently large, but at the same time we need to implement on server's quantum computer the unitary corresponding to this function. Therefore, given the current limited number of available qubits, we choose a 2-regular function that can be easily inverted by brute-force attacks. The scope of this implementation is to demonstrate the correctness and the randomness of the HBC − QFactory protocol.

For the implementation we use the IBM Quantum Experience service and will run all the experiments to prove the correctness and randomness of the protocol using both the simulator "`ibmq_qasm_simulator`" (32 available qubits) and the IBM real devices: "`ibmq_athens`" (5 available qubits) and "`ibmq_melbourne`" (16 available qubits).

7.1. Function Construction for Simulation

We will construct a specific 2-regular function (which given the limitations of the number of available qubits cannot be LWE-based or one-way). More specifically, we define the following 2-regular family of functions $f_{A,B} : \{0,1\}^3 \to \{0,1\}^2$, where the public key is a pair of 2 matrices $A, B \in \{0,1\}^{3\times 3}$:

$$f_{A,B}(x_1x_2x_3) = \left(\bigoplus_{i,j} A_{i,j}x_ix_j\right) \| \left(\bigoplus_{i,j} B_{i,j}x_ix_j\right) \tag{18}$$

where by $\|$ we denote the concatenation of the 2 bits.

To construct the Key Generation algorithm, we will proceed as follows. We first sample uniformly at random the trapdoor information, $(d_0, e) \in \{0,1\}^2$, and we will construct the public key (A, B) as a function of the trapdoor in the following way:

$$A_{e+1,e+1} = A_{2-e,3} = B_{3,3} = 1, B_{2-e,2-e} = d_0, \text{ and all others elements are } 0 \tag{19}$$

For the Inversion algorithm, we proceed in the following way:

Given a function image $y = (y_1, y_2) \in \{0,1\}^2$, we compute its 2 preimages $x = (x_1, x_2, x_3) \in \{0,1\}^3$ and $x' = (x'_1, x'_2, x'_3) \in \{0,1\}^3$ as:

$$\begin{aligned} &x_{2-e} = 0,\ x_{1+e} = y_1,\ x_3 = y_2 \\ &x'_{2-e} = 1,\ x'_{1+e} = y_1 \oplus y_2 \oplus d_0,\ x'_3 = y_2 \oplus d_0 \end{aligned} \tag{20}$$

We emphasize that by constructing f in this particular way, we have the structure required for both HBC − QFactory and Malicious QFactory functions. Indeed, if we define the "hardcore predicate" $h(x_1x_2x_3) = x_3$ (as in the Malicious QFactory protocol), then the two preimages x and x' of any $y \in \{0,1\}^2$ are such that $h(x) \oplus h(x') = d_0$.

Moreover, the circuit to implement the unitary $U_{f_{A,B}}$, it is simple to derive from A and B as we just need to implement a Toffoli gate (or CNOT if $i = j$) between i-th input qubit, j-th input qubit and the first output qubit whenever $A_{ij} = 1$, and similarly with the second output qubit whenever $B_{ij} = 1$.

7.2. Results of Implementation of HBC − QFactory

7.2.1. Randomness

We first want to indicate the randomness of the output of the HBC − QFactory protocol. By this we mean that we want to show that the quantum output is a $|+_\theta\rangle$ state, where θ is uniformly sampled from a set of 8 possible values: $\{0, \frac{\pi}{8}, \cdots, \frac{7\pi}{8}\}$.

The following plot indicates the distribution of θ, can be seen in Figure 3.

We run the protocol 5000 number of times, where the distribution is taken over the randomness of the 2 measurements: y (image register) and b (the final measurement—the preimage register) and over the uniform distribution of the measurement angles α and the plot in Figure 3 indicated an almost uniform distribution of θ. We want to emphasize that the number of trial runs was chosen as the largest choice we could accommodate on the IBM Quantum Service, as a higher number of runs resulted in different failures of obtaining a result from the quantum service.

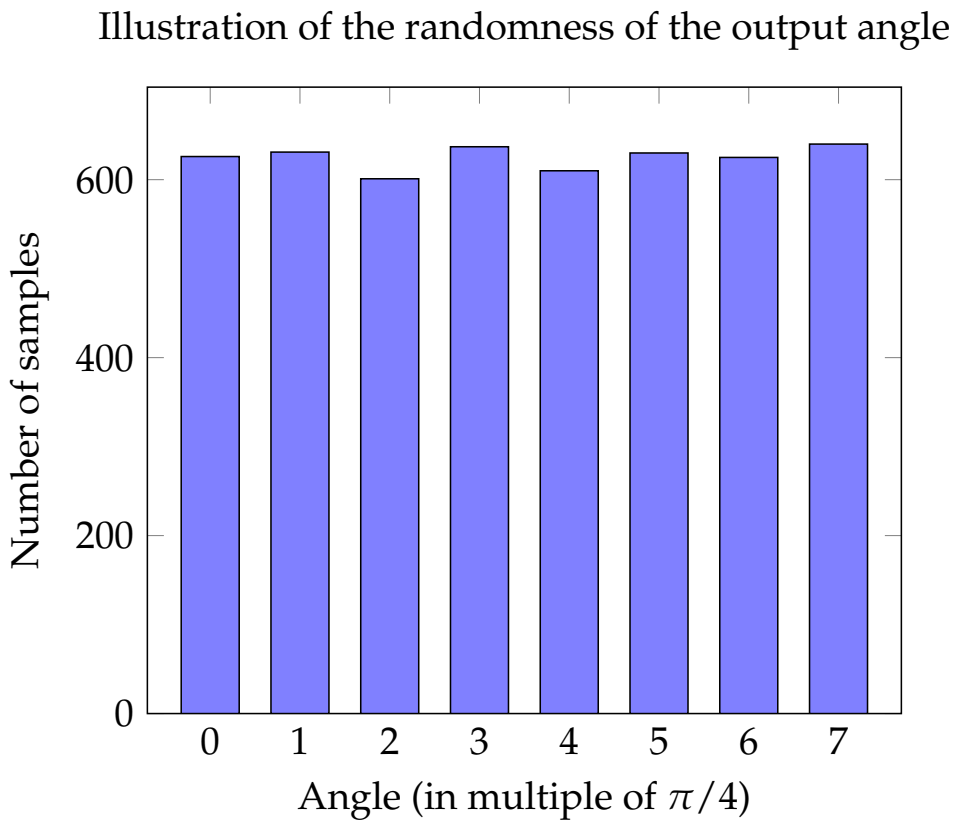


Figure 3. 5000 runs of HBC − QFactory using a fixed function $f_{A,B}$ described in Section 7.1. The plot depicts the resulting distribution, illustrating the randomness of the output distribution.

7.2.2. Correctness

For the correctness of the protocol, we want to show 2-fold:

- The server always obtains a $|+_\theta\rangle$ type of state on his side;
- the client, by knowing the preimages of each image y, can always efficiently compute this θ.

Additionally, we will actually run 2 different types of experiments: using the simulator and a real quantum device. We proceed in the following two different manners.

Interestingly, we proceed in 2 different manners:

When running the simulator we check that the state description the client obtained and the quantum state resulted on the server's side correspond to the same value of θ, in the following manner: For client we run the algorithm described in HBC − QFactory and

we compute the value θ. To check that server obtained exactly the state $|+_\theta\rangle$, the simulator allows us to get the description of the corresponding final quantum output state (as a vector), and we are therefore able to check that it matches perfectly.

Therefore, the first correctness evidence is showed in Figure 4, which depicts the values of θ_A obtained by client and the values of θ_B, where $|+_{\theta_B}\rangle$ is obtained by server. As observed $\theta_A = \theta_B$ for all the runs of our protocol.

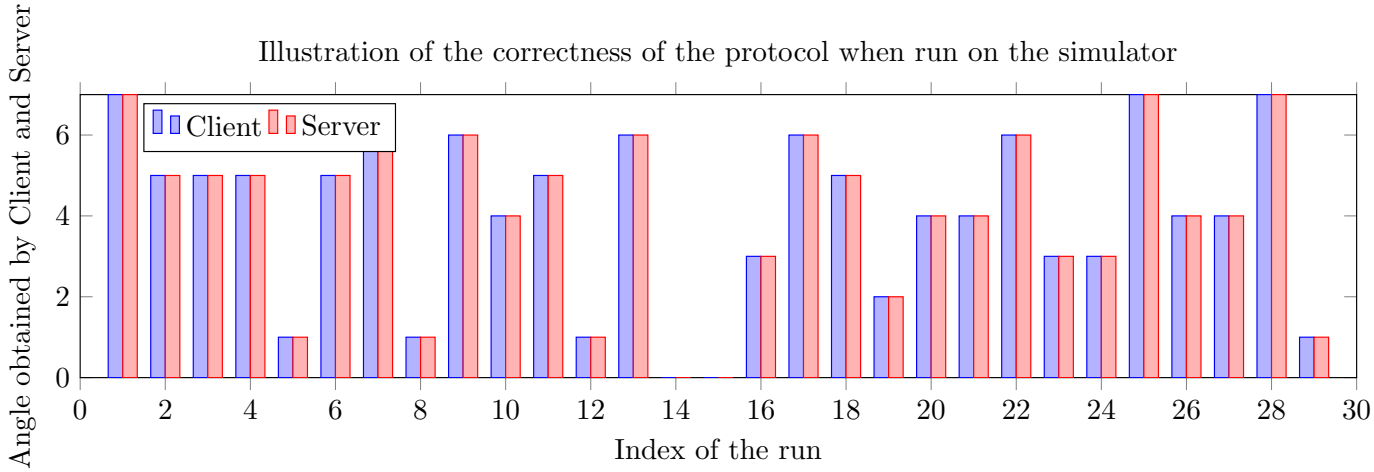


Figure 4. Values of θ_A (written as multiples of $\frac{\pi}{4}$) obtained by client and the corresponding θ_B obtained by server for 29 different runs of HBC − QFactory on the simulator, illustrating the correctness of the protocol.

For the run on the actual quantum device, client obtains the θ in the same way. However, we are no longer able to get the vector description of the quantum state of server, because now it corresponds to an actual physical qubit. Now, the only way to see what state server has on his side is to perform a final measurement on the output qubit $|\psi_{out}\rangle$.

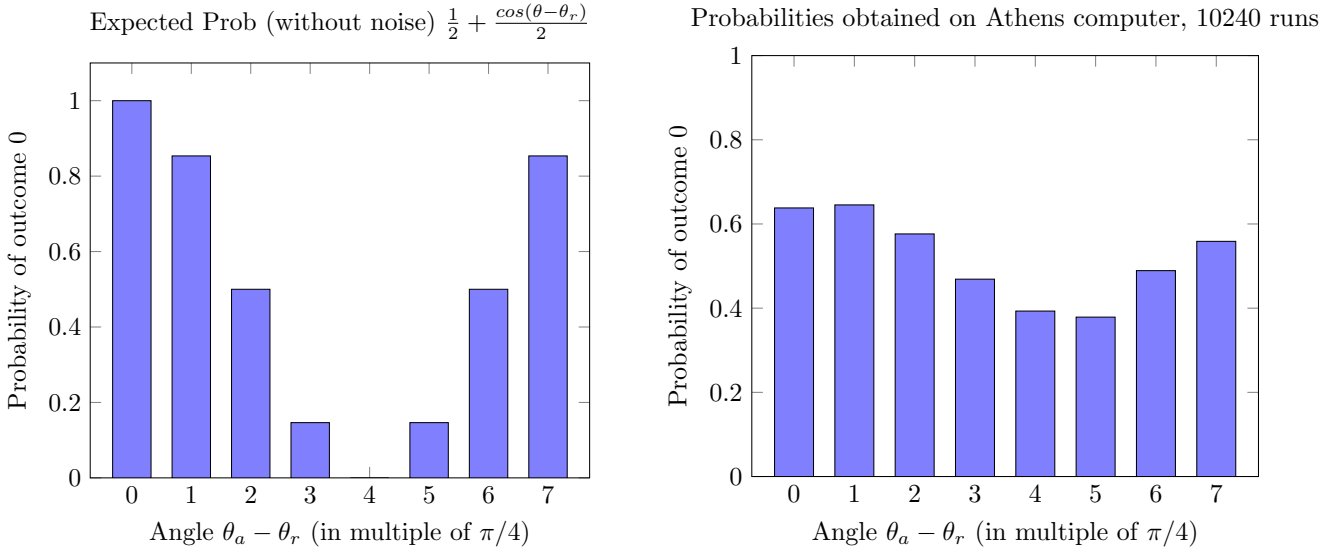
One way would be to measure the output state $|\psi_{out}\rangle$ in the $\{|+_\theta\rangle, |-_\theta\rangle\}$ basis, where θ would be the angle obtained by client through her classical computations, and in this way for correctness we should have always outcome 1 for this measurement.

However, this is not possible, because the IBM does not allow for intermediate measurements. Therefore, the way we will proceed for this is the following:

At the beginning of the protocol, we pick a random θ_r, and then we measure the final state of server $|\psi_{out}\rangle$ in the basis $\{|+_{\theta_r}\rangle, |-_{\theta_r}\rangle\}$. Then, once we receive all the measurement outcomes (preimage, image registers and $|\psi_{out}\rangle$) we first compute (using client's algorithm) the actual θ and then we check:

- If $\theta = \theta_r$, we should obtain measurement outcome 0 with probability 1;
- if $\theta = \theta_r \pm \pi$ we should obtain 1 with probability 1;
- if $\theta = \theta_r \pm \frac{\pi}{2}$ we should get 1 with probability $\frac{1}{2}$;
- if $\theta = \theta_r \pm \frac{\pi}{4}$ we should get 0 with probability $\frac{1}{2} + \frac{\sqrt{2}}{4}$;
- and, in general, the probability of the outcome 0 is equal to: $p(0) = \frac{1}{2} + \frac{cos(\theta - \theta_r)}{2}$.

Therefore, in Figure 5a, we can see the expected probability of the measurement outcome to be 0-which corresponds to the "ideal" quantum computer case. Then, in Figure 5b we have the actual probability of outcome 0, as a result of running it on the "noisy" real quantum device—which corresponds to the real run.



(**a**) (**b**)

Figure 5. (**a**) Ideal case: θ is the honest angle expected by client, and θ_r is the random angle used to measure the state obtained by server. (**b**) Real case: θ is the honest angle expected by client, and θ_r is the random angle used to measure the state obtained by server.

8. Conclusions

8.1. Summary of Results and Discussion

The current work studies the problem of secure delegation of quantum computations between a fully classical honest client and a quantum untrusted server. To achieve this, we introduce the classical client remote state preparation ($\mathsf{CC}-\mathsf{RSP}$) primitive, whose purpose is to remove the need for quantum communication in quantum computation and communication protocols and replace it with classical communication and post-quantum security. Our protocol denoted as $\mathsf{HBC}-\mathsf{QFactory}$ allows a fully classical party (client) to instruct the preparation of a sequence of random quantum states on some distant party (server) in a way that the description of the produced quantum states is known to the client but remains hidden from the server. This protocol relies on the existence of a cryptographic family of functions, specifically a trapdoor one-way function that is quantum-safe, two-regular and collision resistant. We prove the security of our construction against an honest-but-curious server, that follows the protocol specifications, but can try to learn about the secret from the classical transcripts. We show that the description of the output qubit is completely hidden from the server using an argument similar to the Goldreich–Levin theorem. Furthermore, to complete the picture, we show how to construct the required cryptographic functions, by giving an explicit family of functions and an instantiation of its parameters that satisfy all the required properties. These functions are based on the Learning-With-Errors problem. Finally, we provide an experiment of this $\mathsf{CC}-\mathsf{RSP}$ protocol on IBM's quantum computer using a toy function. This is the first proof-of-principle experiment of classical client remote state preparation.

The main motivation for the current work is that the $\mathsf{CC}-\mathsf{RSP}$ primitive and particularly the $\mathsf{HBC}-\mathsf{QFactory}$ construction, can be used by classical clients to replace a specific quantum channel and enable them to securely delegate quantum computations, for example, by combining $\mathsf{HBC}-\mathsf{QFactory}$ with the protocol of [21]. However, we have noted that enabling classical parties to participate in quantum communication protocols opens numerous opportunities for the applications outlined in Section 1.3 and Appendix A. Indicatively, our protocol can be used in: quantum money, quantum coin-flipping, quantum signatures, etc. The classical client remote state preparation has also been recently

employed to construct a quantum two-party computation between a quantum party and a fully classical party [41].

At this point we would like to discuss the practical setup advantages and limitations. The protocol we consider has two main aspects that are evaluated: security (maintaining the privacy of the classical description of the remote state prepared) and correctness (ensuring that an honest server produces the desired state in his lab). The fact that the client is fully classical and there is no quantum state prepared by the (honest) client means that any imperfections and noise could exist only on the server's side. Since the server is the adversary, imperfections on his side only deteriorate his chances of cheating and thus the security of the protocol is not affected from such practical considerations. In other words, noise does not help the server to obtain further information on the classical description of the ideal state that is prepared. On the other hand, the correctness of the protocol is affected by noise as we have seen in our simulations in Section 7. This serves as a reminder, that our setting is more relevant when the server has the ability to perform universal (fault-tolerant) quantum computations. For example, attempts to use such protocol in Noisy Intermediate Scale Quantum (NISQ) devices, is likely to lead to very noisy output qubits and thus, $\mathsf{HBC-QFactory}$ is not really practically implementable in this setting.

8.2. Future Directions

Given the need for a practical solution for the classical secure delegation of quantum computations problem in order to exploit the full potential of quantum computers, an important priority is to reduce the cost and bring this primitive close to practice. There are a number of potential improvements to consider such as: finding trapdoor functions that lead to smaller overhead (either by using a different function construction, or a different implementation or a different "quantum-hard" problem) and making a more robust version that can tolerate noise on the server's side, while maintaining the same security guarantees.

We have, somehow heuristically, mentioned that our primitive (classical client remote state preparation) can be combined with other protocols to achieve a number of applications. Another future direction is to make this intuition concrete and prove that those applications that could use our primitive as a subroutine, are secure (as it was done in the quantum multiparty computation setting [41]). The formal (and not always achievable) way to deal with composite protocols is to use a composable security framework. In the recent work of [20], the use of a classical client remote state preparation in bigger contexts has been studied using such a composable security framework, and an impossibility result for composable $\mathsf{CC-RSP}$ was proven. This implies that we cannot resort to such an "economic" solution, where one proves the security of the primitive and can then use it directly as subroutine. Instead, for each of the applications we outlined that would use $\mathsf{CC-RSP}$ as a sub-module, we have to prove the security of the resulting protocol afresh using a game-based security definition. This is an important future direction. Note, that in [20], this game-based approach was used for proving the security of a classical secure delegation of quantum computations protocol (combining a version of $\mathsf{CC-RSP}$ with the protocol of [21]).

Author Contributions: All authors contributed in the conception, proofs and writing of the paper. A.C. and L.C. performed the simulations, E.K. secured funding and E.K. and P.W. supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by EPSRC grants EP/N003829/1, EP/M013243/1, EP/T001062/1 and FA9550-17-1-0055.

Acknowledgments: A.C. and P.W. are very grateful to Thomas Zacharias, Aggelos Kiayias and especially Yiannis Tselekounis for many useful discussions about the security proofs. L.C. also thanks Atul Mantri and Dominique Unruh for useful preliminary conversations, with a special mention to Céline Chevalier whose discussions were a precious source of inspiration. A.C. would also like to show his appreciation to his grandmother, Petra Ilie for all the support and help she has given to him his entire life.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. $\mathsf{CC}-\mathsf{RSP}_\theta$ within Several Applications

In Section 1.3 we listed several applications that can use the $\mathsf{CC}-\mathsf{RSP}_\theta$ functionality to allow for fully classical parties to participate using, a potentially malicious, quantum server. Here we give details on how to use the exact output of our QFactory protocol in these applications. We emphasize that in all protocols in which the "server" used by the classical party is a malicious party, the cost of using our QFactory construction is that the security becomes computational and applies in the quantum-honest-but-curious setting.

1. In the quantum homomorphic encryption scheme AUX in [4], where the target quantum computation must have constant T-gate depth, using our QFactory protocol would allow a classical client to participate (delegate such computation) provided, of course, that the input/output are classical. Specifically, as the input is classical, the client will instruct the server to prepare a quantum state of the classical one-time pad of this input (and then the client will also send to the server a classical homomorphic encryption of the classical one-time pad key of each of the input's bits). Moreover, for every T-gate in the quantum computation, the auxiliary qubits in the evaluation key can be produced using QFactory: $\{|+\rangle, P|+\rangle = |+_{2\pi/4}\rangle, Z|+\rangle = |+_{4\pi/4}\rangle, ZP|+\rangle = |+_{6\pi/4}\rangle\}$.
   We note that due to the use of a classical fully homomorphic encryption scheme, the AUX protocol [4] has computationally security, thus, the computational security offered by the QFactory is not downgrading the security of this protocol.
2. In the blind delegated quantum computation protocol of [21], the client needs to prepare and send to the server qubits, randomly chosen, from the set of states $\{|+\rangle, |+_{\pi/4}\rangle, ..., |+_{7\pi/4}\rangle\}$. This is exactly the set of states of Equation (3) which are given by the QFactory. It follows that our construction eliminates the need for quantum communication and thus any classical client can use this protocol.
3. The verifiable blind quantum computation protocol in [32], the only quantum ability that the verifier needs is to prepare and send to the prover single qubits, randomly chosen, from the set of states $\{|+_{k\pi/4}\rangle\}$. Again, this is exactly the set of states given by the QFactory. Therefore, the quantum communication, and thus quantum abilities of the verifier, can be completely replaced by the $\mathsf{CC}-\mathsf{RSP}$ primitive.
4. For the quantum key-distribution construction in [33], we can use two conjugate bases to realise this protocol, namely: the diagonal basis $\{|+\rangle, |+_\pi\rangle\}$ and the left-right handed circular basis $\{|+_{\pi/2}\rangle, |+_{3\pi/2}\rangle\}$. All these four quantum states can be obtained by the QFactory protocol. We note that if one was interested in obtaining exactly and only this set of states, we can modify the QFactory to do so, in a way that actually simplifies the proofs too. For example, we could simply ask to measure the qubits in the second stage in the basis $\{|\pm_{\alpha_i\pi/2}\rangle\}$. As the quantum coin flipping protocol of [33], the quantum money protocol of [34] or the quantum digital signatures protocol of [36] only require, as in [33], any pair of conjugate bases, this implies that we can use QFactory in a straight forward way. On the other hand, for the quantum coin flip construction in [35], the single qubit quantum states needed are of the form $\sqrt{a}|0\rangle + (-1)^{\alpha_i}\sqrt{1-a}|1\rangle$, which might be achieved by a different construction of the $\mathsf{CC}-\mathsf{RSP}_\theta$.
5. In the multiparty quantum computation protocol of [39], the n clients need to send multiple copies of quantum states in the set $\{|+_{k\pi/4}\rangle\}$ to the server, who entangles and measures them all but one. Using QFactory all these states will be prepared by the server, which would enable the n clients to be fully classical.
6. The verifiable blind quantum computation protocols in [30,31] or the two-party quantum computation protocols in [37,38], require the honest party to prepare single qubit states from the set of states $\{|0\rangle, |1\rangle, |+_{k\pi/4}\rangle\}$. While the QFactory primitive can output the $|+_{k\pi/4}\rangle$ states, in order to make the honest party fully classical, we

need to change the construction of QFactory in order to also be able to output the $|0\rangle$ and $|1\rangle$ states, and maintain the same guarantees in privacy as in the QFactory.

Appendix B. Full Proof of Theorem 7

Proof. From Equation (13), we have the definition of $\widetilde{B}$ in terms of the three corresponding bits and we aim to prove that it is hard-core, i.e., that Equation (15) is satisfied. We will follow the five steps outlined in the main text. Before that let us define some simple identities that will be used. For any $a, b, d, e \in \mathbb{N}$, we have:

$$(a+b) \bmod 8 = (a \bmod 8 + b \bmod 8) \bmod 8 \tag{A1}$$

$$[(a+b) \bmod 8] \bmod 4 = (a \bmod 4 + b \bmod 4) \bmod 4 \tag{A2}$$

$$[(a+b) \bmod 4] \bmod 2 = (a \bmod 2 + b \bmod 2) \bmod 2 \tag{A3}$$

$$(2a) \bmod 4 = 2 \cdot (a \bmod 2) \tag{A4}$$

$$(2a) \bmod 8 = 2 \cdot (a \bmod 4) \tag{A5}$$

$$(2d+e) \bmod 4 - e \bmod 2 = [2d + e - (e \bmod 2)] \bmod 4 \tag{A6}$$

$$(2d+e) \bmod 8 - e \bmod 2 = [2d + e - (e \bmod 2)] \bmod 8 \tag{A7}$$

We now return to Equation (13)

$$\widetilde{B} = g(x - x') = \sum_{i=1}^{n} (x_i - x'_i)(4b_i + \alpha_i) \bmod 8$$

where $\widetilde{B} = \widetilde{B_1}\widetilde{B_2}\widetilde{B_3}$, with $\widetilde{B_j} \in \{0,1\}$. We also define $\widetilde{x} = x \oplus x' \in \{0,1\}^n$ and $z \in \{-1,0,1\}^n$ be the vector defined as: $z_i = x_i - x'_i = (-1)^{x'_i}\widetilde{x}_i$, $\forall\, i \in \{1,2,...,n\}$.

Step 1: We will rewrite this expression in terms of single bits and obtain the expression of Equation (15). We have $g(z) = \sum_{i=1}^{n} z_i(4b_i + \alpha_i) \bmod 8$, or equivalently:

$$4\widetilde{B_1} + 2\widetilde{B_2} + \widetilde{B_3} = \left[\left(4 \sum_{i=1}^{n} z_i b_i \right) \bmod 8 + \left(\sum_{i=1}^{n} z_i \alpha_i \right) \bmod 8 \right] \bmod 8$$

We define the following terms: $\alpha_i = 4\alpha_i^{(1)} + 2\alpha_i^{(2)} + \alpha_i^{(3)}$, where $\alpha_i^{(1)}, \alpha_i^{(2)}, \alpha_i^{(3)}$ are the 3 bits of α_i and $\alpha^{(j)} \in \{0,1\}^n$ are the vectors consisting of the j-th bit of all values α_i, $\forall i \in \{1,2,...,n\}, j \in \{1,2,3\}$;

$$S_0 = \sum_{i=1}^{n} z_i b_i \quad ; \quad S_1 = \sum_{i=1}^{n} z_i \alpha_i^{(1)}$$
$$S_2 = \sum_{i=1}^{n} z_i \alpha_i^{(2)} \quad ; \quad S_3 = \sum_{i=1}^{n} z_i \alpha_i^{(3)}$$

We also notice that under mod 2, we have that:

$$S_j \bmod 2 = \sum_{i=1}^{n} \widetilde{x}_i \alpha_i^{(j)} \bmod 2 = \langle \widetilde{x}, \alpha^{(j)} \rangle \bmod 2, \text{ for } j \in \{1,2,3\}.$$

Then, we have:

$$4\widetilde{B_1} + 2\widetilde{B_2} + \widetilde{B_3} = (\sum_{i=1}^{n} (x_i - x'_i)(4b_i + \alpha_i)) \bmod 8 =$$

$$\left[4S_0 + \sum_{i=1}^{n}(x_i - x'_i)(4\alpha_1^{(i)} + 2\alpha_2^{(i)} + \alpha_3^{(i)})\right] \bmod 8$$

$$4\widetilde{B_1} + 2\widetilde{B_2} + \widetilde{B_3} = (4S_0 + 4S_1 + 2S_2 + S_3) \bmod 8 \tag{A8}$$

Applying mod 2 to Equation (A8), we get:

$$\widetilde{B_3} = (4S_0 \bmod 2 + 4S_1 \bmod 2 + 2S_2 \bmod 2 + S_3 \bmod 2) \bmod 2$$

$$\boxed{\widetilde{B_3} = S_3 \bmod 2 = \langle \widetilde{x}, \alpha^{(3)} \rangle \bmod 2} \tag{A9}$$

If, instead we apply mod 4 to Equation (A8), we get:

$$\begin{aligned} 2\widetilde{B_2} + \widetilde{B_3} &= [4S_0 \bmod 4 + 4S_1 \bmod 4 + 2S_2 \bmod 4 + S_3 \bmod 4] \bmod 4 \\ 2\widetilde{B_2} + \widetilde{B_3} &= [(2S_2) \bmod 4 + S_3 \bmod 4] \bmod 4. \text{ Using (A4), we have:} \\ 2\widetilde{B_2} + \widetilde{B_3} &= [2(S_2 \bmod 2) + S_3 \bmod 4] \bmod 4 \end{aligned}$$

$$\begin{aligned} \widetilde{B_2} &= \frac{1}{2}\{[2(S_2 \bmod 2) + S_3 \bmod 4] \bmod 4 - S_3 \bmod 2\}. \text{ Using (A6):} \\ \widetilde{B_2} &= \frac{1}{2}[2(S_2 \bmod 2) + S_3 \bmod 4 - S_3 \bmod 2] \bmod 4 \\ \widetilde{B_2} &= \frac{1}{2}\left\{2 \cdot \left[(S_2 \bmod 2) + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right]\right\} \bmod 4. \text{ Using (A4), we obtain:} \\ \widetilde{B_2} &= \left[S_2 \bmod 2 + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right] \bmod 2. \\ \widetilde{B_2} &= S_2 \bmod 2 \oplus \left(\frac{S_3 \bmod 4 - S_3 \bmod 2}{2}\right) \end{aligned}$$

$$\boxed{\widetilde{B_2} = \langle \widetilde{x}, \alpha^{(2)} \rangle \bmod 2 \oplus \left(\frac{S_3 \bmod 4 - S_3 \bmod 2}{2}\right)} \tag{A10}$$

Finally, we can derive $\widetilde{B_1}$:

$$\begin{aligned} \widetilde{B_1} = \tfrac{1}{4}\{&(4S_0 + 4S_1 + 2S_2 + S_3) \bmod 8 - (S_3 \bmod 2) - \\ &2\left[\left(S_2 \bmod 2 + \tfrac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\} \end{aligned}$$

Using (A7):

$$\begin{aligned} \widetilde{B_1} = \tfrac{1}{4}\{&(4S_0 + 4S_1 + 2S_2 + S_3 - (S_3 \bmod 2)) \bmod 8 - \\ &2\left[\left(S_2 \bmod 2 + \tfrac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\} \end{aligned}$$

Using (A5):

$$\begin{aligned} \widetilde{B_1} = \tfrac{1}{4}\Big\{&2\left[\left(2S_0 + 2S_1 + S_2 + \tfrac{S_3 - S_3 \bmod 2}{2}\right) \bmod 4\right] - \\ &2\left[\left(S_2 \bmod 2 + \tfrac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]\Big\} \end{aligned}$$

$$\widetilde{B_1} = \tfrac{1}{2}\left[\left(2S_0 + 2S_1 + S_2 + \tfrac{S_3 - S_3 \bmod 2}{2}\right) \bmod 4 - \left(S_2 \bmod 2 + \tfrac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right) \bmod 2\right]$$

Using (A6) we can rewrite the first term, and we get:

$$\begin{aligned} \widetilde{B_1} = \frac{1}{2}\Big\{&\left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2 + \left[2(S_0 + S_1) + S_2 + \frac{S_3 - S_3 \bmod 2}{2} - \right. \\ &\left. - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2\right] \bmod 4 - \left[S_2 \bmod 2 + \frac{(S_3 \bmod 4 - S_3 \bmod 2)}{2}\right] \bmod 2\Big\} \end{aligned}$$

Combining the first and third term:

$$\begin{aligned}\widetilde{B_1} &= \frac{1}{2}\left\{\left[(S_2 - S_2 \bmod 2) + \frac{S_3 - S_3 \bmod 4}{2}\right] \bmod 2 + \right. \\ &\left. + \left[2(S_0+S_1) + S_2 + \frac{S_3 - S_3 \bmod 2}{2} - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2\right] \bmod 4\right\}\end{aligned}$$

We notice that both $S_2 - S_2 \bmod 2$ and $\frac{S_3 - S_3 \bmod 4}{2}$ are even, so the first big term is 0:

$$\widetilde{B_1} = \frac{1}{2}\left\{\left[2(S_0+S_1) + S_2 + \frac{S_3 - S_3 \bmod 2}{2} - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2\right] \bmod 4\right\}$$

which can be rewritten as:

$$\widetilde{B_1} = \frac{1}{2}\left\{\left\{2 \cdot \left[S_0 + S_1 + \frac{\left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2}{2}\right]\right\} \bmod 4\right\}$$

Finally, using (A4), we get:

$$\begin{aligned}\widetilde{B_1} &= \left[S_0 + S_1 + \frac{\left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2 \\ \widetilde{B_1} &= S_1 \bmod 2 \oplus S_0 \bmod 2 \oplus \left[\frac{\left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2\end{aligned}$$

$$\boxed{\widetilde{B_1} = \langle \widetilde{x}, \alpha^{(1)} \rangle \bmod 2 \oplus \langle \widetilde{x}, b \rangle \bmod 2 \oplus \left[\frac{\left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) - \left(S_2 + \frac{S_3 - S_3 \bmod 2}{2}\right) \bmod 2}{2}\right] \bmod 2} \quad \text{(A11)}$$

Important observation: $\widetilde{B_1}$, $\widetilde{B_2}$, $\widetilde{B_3}$ all depend on the same value of x and x' (or $\widetilde{x}$, or z). Therefore, to make our analysis easier, we can consider that z and $\widetilde{x}$ are fixed. Then, if we define the function:

$$B(r) = \langle \widetilde{x}, r \rangle \bmod 2 = \left(\sum_{i=1}^{n} \widetilde{x}_i r_i\right) \bmod 2 \quad \text{(A12)}$$

we can rewrite $\widetilde{B_3}, \widetilde{B_2}, \widetilde{B_1}$ as in Equation (15) completing Step 1:

$$\begin{aligned}\widetilde{B_3} &= B\left(\alpha^{(3)}\right) \\ \widetilde{B_2} &= B\left(\alpha^{(2)}\right) \oplus h_2(z, \alpha^{(3)}) \\ \widetilde{B_1} &= B\left(\alpha^{(1)}\right) \oplus h_1(z, \alpha^{(3)}, \alpha^{(2)}, b)\end{aligned} \quad \text{(A13)}$$

where:

$$h_2(z, \alpha^{(3)}) := \frac{\langle z, \alpha^{(3)} \rangle \bmod 4 - \langle z, \alpha^{(3)} \rangle \bmod 2}{2} \quad \text{(A14)}$$

$$\begin{aligned}h_1(z, \alpha^{(3)}, \alpha^{(2)}, b) &:= \langle z, b \rangle \bmod 2 \ \oplus \\ \oplus \left[\frac{\left(\langle z, \alpha^{(2)} \rangle + \frac{\langle z, \alpha^{(3)} \rangle - \langle z, \alpha^{(3)} \rangle \bmod 2}{2}\right) - \left(\langle z, \alpha^{(2)} \rangle + \frac{\langle z, \alpha^{(3)} \rangle - \langle z, \alpha^{(3)} \rangle \bmod 2}{2}\right) \bmod 2}{2}\right] &\bmod 2\end{aligned} \quad \text{(A15)}$$

Step 2: We see from Equation (A11) that each of the three bits involve a term similar to that of the GL Theorem 2 (the $B\left(\alpha^{(i)}\right)$ term), but with two the important differences. First,

there is another term, and the bits of $\widetilde{B}$ are XORs of the GL-looking term and that other one. The second type of terms (that involve h_1, h_2) depend on variables that appear in the expressions of other bits, potentially introducing correlations among the different bits. We will deal with the issue of correlations in Step 3, while with the effects of having extra terms in Steps 4 and 5. Here we deal with the second important difference, namely that the GL-looking terms (those of the form $\langle \tilde{x}, r \rangle$ mod 2) depend on $\tilde{x}$ rather than x in the inner product. For the remaining Step 2, we assume that the first issue is resolved and it all reduces to GL theorem subject to having $\tilde{x}$ rather than x.

Since we have $\tilde{x}$ in our expression, if we follow the same proof with that of the GL theorem we can follow the proof until the point that we end up with obtaining a polynomial number of guesses for $\tilde{x}$ of which one is the correct value with probability negligibly close to unity. Now to continue with the proof we are lacking two elements. First, in GL theorem they use the fact that computing $f(x)$ given x is easy, and check one-by-one the polynomial guesses to see which one (if any) is correct. We cannot do this since we only obtain $\tilde{x}$ and there is no way with no extra information to check if $\tilde{x}$ actually corresponds to a given image $y = f(x) = f(x')$. The second issue, is that even if we could check this, having obtained $\tilde{x}$ does not contradict the definition of one-way function (Definition 2).

We resolve both these issues with two observations. **Observation 1:** We notice that because of the 2-regularity property of f, $\widetilde{x}$ is uniquely determined by x ($f(x) = f(x')$, $\widetilde{x} = x \oplus x'$). **Observation 2:** The assumption that our 2-regular trapdoor function f is second preimage resistant (i.e., a QPT adversary given x, cannot find the second preimage x', where $f(x) = f(x'))$ means that:

$$\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x' \text{ such that } f(x) = f(x')] \leq \mathsf{negl}(n)$$

As

$$\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x'] = \Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = x' \oplus x]$$

we have that:

$$\Pr_{x \leftarrow \{0,1\}^n} [\mathcal{A}(1^n, x) = \widetilde{x}] \leq \mathsf{negl}(n) \tag{A16}$$

As we have mentioned, following the GL theorem proof we would obtain polynomially many guesses $\tilde{x}_g$ for $\tilde{x}$ (where subscript g stands for guess). Now, by the second preimage resistance, if we are given x we should be unable to obtain x' in polynomial time. However, using our polynomially many guesses for $\tilde{x}$ and checking for each guess if $f(x \oplus \tilde{x}_g) = f(x)$, we can obtain with probability negligible close to unity the correct $\tilde{x}$ and therefore come to contradiction with Equation (A16).

Step 3: Since the different bits involve common variables, to prove that our function is hard-core we need to consider the issue of correlations. One way to deal with this would be to prove the independence of both the bits and of the optimal guessing algorithms. We, instead, use the Vazirani–Vazirani Theorem 3, which for our case means that it suffices to show that: $\widetilde{B_1}$, $\widetilde{B_2}$, $\widetilde{B_3}$, $\widetilde{B_1} \oplus \widetilde{B_2}$, $\widetilde{B_1} \oplus \widetilde{B_3}$, $\widetilde{B_2} \oplus \widetilde{B_3}$, $\widetilde{B_1} \oplus \widetilde{B_2} \oplus \widetilde{B_3}$ are all hard-core predicates for f.

The most general expression that captures all these hard-core predicates (formed from the subsets of $\{\widetilde{B_1}, \widetilde{B_2}, \widetilde{B_1}\}$) is:

$$E(x, r_1, r_2, r_3) = \langle \widetilde{x}, r_1 \oplus r_2 \rangle \text{ mod } 2 \oplus g(z, r_2, r_3) \tag{A17}$$

where g can be any binary function. Using $\langle \widetilde{x}, r_1 \oplus r_2 \rangle \bmod 2 = \langle \widetilde{x}, r_1 \rangle \bmod 2 \oplus \langle \widetilde{x}, r_2 \rangle \bmod 2$ we can rewrite this as:

$$E(x, r_1, r_2, r_3) = \langle \widetilde{x}, r_1 \rangle \bmod 2 \oplus g'(z, r_2, r_3) \tag{A18}$$

where $g'(z, r_2, r_3) = \langle \widetilde{x}, r_2 \rangle \bmod 2 \oplus g(z, r_2, r_3)$. In other words, in order to prove that $\widetilde{B_1}\widetilde{B_2}\widetilde{B_3}$ is a hard-core function for f, it suffices to prove that $E(x, r_1, r_2, r_3)$ is a hard-core predicate for f.

Step 4: In this step, we will see how we can effectively fix all but one variables, and turn Equation (A18) to depend only on r_1.

We want to prove that if there exists a QPT algorithm $\mathcal{A}$ that can guess the predicate E as given in Equation (A18), $\mathcal{A}(f(x), r_1, r_2, r_3) = E(x, r_1, r_2, r_3)$ with probability non-negligible better than $1/2$, then the second preimage resistance assumption is violated by constructing a QPT algorithm $\mathcal{A}'$ that, when given x can obtain $\widetilde{x}$, $\mathcal{A}'(f(x), 1^n) = \widetilde{x}$, with non-negligible probability.

We now assume that the advantage $\mathcal{A}$ has in computing is $\varepsilon(n)$, without restricting $\varepsilon(n)$ to be non-negligible, aiming to reach a contradiction if this $\varepsilon(n)$ is inverse polynomial. We therefore assume:

$$\Pr_{\substack{x \leftarrow \{0,1\}^n \\ r_1 \leftarrow \{0,1\}^n \\ r_2 \leftarrow \{0,1\}^n \\ r_3 \leftarrow \{0,1\}^n}} [\mathcal{A}(f(x), r_1, r_2, r_3) = E(x, r_1, r_2, r_3)] = \frac{1}{2} + \varepsilon(n) \tag{A19}$$

Since the different variables (x, r_1, r_2, r_3) are chosen randomly and independently, we can effectively "fix" one variable. We can consider the set of values of that variable that satisfy some condition that we need and name these values "Good" values (e.g., the guessing algorithm $\mathcal{A}$ to succeed with higher than negligible probability). Then we can work with the assumption that the fixed variable is within the "Good" set, with the only caveat that at the end, whatever probability of inversion we obtain, is conditional on the fixed variables being "Good" and thus we need to multiply that probability with the probability that the fixed variable is "Good". For this reason, it is important that the probability of being "Good" (ratio between cardinality of Good values and total number of values) should be at least inverse polynomial.

We will, therefore, be using the following Lemma:

Lemma A1. *Let* $\Pr_{(v_1, \cdots, v_k) \leftarrow \{0,1\}^n \times \cdots \times \{0,1\}^n} [Guessing] \geq p + \varepsilon(n)$*, then for any variable* v_i*, there exists a set* $Good_{v_i} \subseteq \{0,1\}^n$ *of size at least* $\frac{\varepsilon(n)}{2} 2^n$*, such that for all* $v_i \in Good_{v_i}$*, we have:*

$$\Pr_{(v_1, \cdots, \not{v_i}, \cdots, v_k) \leftarrow \{0,1\}^n \times \cdots \times \{0,1\}^n} [Guessing] \geq p + \frac{\varepsilon(n)}{2}$$

where the latter probability is taken over all variables except v_i*.*

Proof.

$$\begin{aligned} p + \varepsilon(n) &\leq \tfrac{1}{2^n} \textstyle\sum_{v_i \in Good_{v_i}} \Pr_{(v_1, \cdots, \not{v_i}, \cdots, v_k) | v_j \in \{0,1\}^n} [Guessing] + \\ &\tfrac{1}{2^n} \textstyle\sum_{v_i \notin Good_{v_i}} \Pr_{(v_1, \cdots, \not{v_i}, \cdots, v_k) | v_j \in \{0,1\}^n} [Guessing] \\ &\leq \tfrac{1}{2^n} |Good_{v_i}| + \tfrac{1}{2^n} \textstyle\sum_{v_i \notin Good_{v_i}} (p + \frac{\varepsilon(n)}{2}) \end{aligned} \tag{A20}$$

$$\begin{aligned} p+\varepsilon(n) &\leq \frac{1}{2^n}|Good_{v_i}| + (p+\frac{\varepsilon(n)}{2}) \\ \frac{\varepsilon(n)}{2}2^n &\leq |Good_{v_i}| \end{aligned} \tag{A21}$$

□

Now we return to Equation (A19), we fix the set of $Good_x$, the set of inputs x such that:

$$\Pr_{\substack{r_1\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n}} [\mathcal{A}(f(x), r_1, r_2, r_3) = E(x, r_1, r_2, r_3)] \geq \frac{1}{2} + \frac{\varepsilon(n)}{2} \quad \forall x \in Good_x \tag{A22}$$

and using Lemma A1 we have $|Good_x| \geq \frac{\varepsilon(n)}{2}2^n$. Note that fixing x is equivalent with fixing $\tilde{x}$ or z, given the definition of the 2-regular function f. Starting with Equation (A22), we can now fix r_3 (conditional on $x \in Good_x$):

$$\Pr_{\substack{r_1\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}} [\mathcal{A}(f(x), r_1, r_2, r_3) = E(x, r_1, r_2, r_3)] \geq \frac{1}{2} + \frac{\varepsilon(n)}{4}$$
$$\forall x \in Good_x \wedge \forall r_3 \in Good_{r_3} \tag{A23}$$

where using again Lemma A1 we have $|Good_{r_3}| \geq \frac{\varepsilon(n)}{4}2^n$. Finally, we can fix r_2 (conditional on $x \in Good_x$ and $r_3 \in Good_{r_3}$):

$$\Pr_{r_1\leftarrow\{0,1\}^n} [\mathcal{A}(f(x), r_1, r_2, r_3) = E(x, r_1, r_2, r_3)] \geq \frac{1}{2} + \frac{\varepsilon(n)}{8}$$
$$\forall x \in Good_x \wedge \forall r_3 \in Good_{r_3} \wedge \forall r_2 \in Good_{r_2} \tag{A24}$$

and again by Lemma A1 we have $|Good_{r_2}| \geq \frac{\varepsilon(n)}{8}2^n$.

Step 5: In Equation (A24), the only variable is r_1. Using Equation (A18) we can see that given that x, r_2, r_3 are all fixed, $E(x, r_1, r_2, r_3) = \langle \tilde{x}, r_1 \rangle \oplus g'(z, r_2, r_3)$ where $g'(z, r_2, r_3) = c$ is constant. As a result that c is a constant, we can define $\tilde{\mathcal{A}} = \mathcal{A} \oplus c$. Now, we can easily see that:

$$\begin{aligned} &\Pr_{r_1\leftarrow\{0,1\}^n} [\mathcal{A}(f(x), r_1, r_2, r_3) = \langle \tilde{x}, r_1 \rangle \bmod 2 \oplus g'(z, r_2, r_3)] \\ = &\Pr_{r_1\leftarrow\{0,1\}^n} [\tilde{\mathcal{A}}(f(x), r_1, r_2, r_3) = \langle \tilde{x}, r_1 \rangle \bmod 2] \end{aligned}$$

Therefore, using Equation (A24), we obtain

$$\Pr_{r_1\leftarrow\{0,1\}^n} [\tilde{\mathcal{A}}(f(x), r_1, r_2, r_3) = \langle \tilde{x}, r_1 \rangle \bmod 2] \geq \frac{1}{2} + \frac{\varepsilon(n)}{8}$$
$$\forall x \in Good_x \wedge \forall r_3 \in Good_{r_3} \wedge \forall r_2 \in Good_{r_2} \tag{A25}$$

which is exactly the expression in GL theorem. There, one obtains guesses for inversion, i.e., to obtain $\tilde{x}$ with a polynomial in $\varepsilon(n)$ probability of success, given the fixed x, r_2, r_3's. Multiplying this with the probability of actually being in $Good_x$ and $Good_{r_3}$ and $Good_{r_2}$ we obtain another polynomial in $\varepsilon(n)$. This rules out the possibility of $\varepsilon(n)$ being inverse polynomial, since that would break the second preimage resistance. As we have already stated, guessing $\tilde{x}$ with inverse polynomial success probability does not contradict the one-way property of the trapdoor function, but it does contradict the second preimage resistance, since given x and $\tilde{x}$ one can obtain deterministically x'.

Concretely, using GL proof to construct from $\tilde{\mathcal{A}}$, a QPT algorithm $\mathcal{A}'$ that obtains $\widetilde{x}$, $\mathcal{A}'(f(x), 1^n) = \widetilde{x}$ for all inputs $x \in \{0,1\}^n$, when, $x \in Good_x$, $r_3 \in Good_{r_3}$ and $r_2 \in Good_{r_2}$, this algorithm $\mathcal{A}'$ succeeds with probability:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n) = \widetilde{x} \,|\, x \in Good_x \wedge r_3 \in Good_{r_3} \wedge r_2 \in Good_{r_2}]$$

$$\geq \frac{\varepsilon^2(n)}{2(32n+\varepsilon^2(n))} \geq \frac{\varepsilon^2(n)}{2(32n+1)}$$

Then, we have:

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n) = \widetilde{x}] \geq$$

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n) = \widetilde{x} \wedge x \in Good_x \wedge r_3 \in Good_{r_3} \wedge r_2 \in Good_{r_2}] \geq$$

$$\geq \Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n) = \widetilde{x} \,|\, x \in Good_x \wedge r_3 \in Good_{r_3} \wedge r_2 \in Good_{r_2}] \cdot$$

$$\cdot \Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[x \in Good_x \wedge r_3 \in Good_{r_3} \wedge r_2 \in Good_{r_2}]$$

We now see that:

$$\Pr[x \in Good_x \wedge r_3 \in Good_{r_3} \wedge r_2 \in Good_{r_2}] =$$
$$\Pr[r_2 \in Good_{r_2} | r_3 \in Good_{r_3} \wedge x \in Good_x] \cdot \Pr[r_3 \in Good_{r_3} | x \in Good_x] \times \Pr[x \in Good_x]$$

By construction we have $\Pr[x \in Good_x] = \frac{|Good_x|}{2^n}$ and $\Pr[r_3 \in Good_{r_3} | x \in Good_x] = \frac{|Good_{r_3}|}{2^n}$ and $\Pr[r_2 \in Good_{r_2} | r_3 \in Good_{r_3} \wedge x \in Good_x] = \frac{|Good_{r_2}|}{2^n}$, which leads to

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n) = \widetilde{x}] \geq \frac{\varepsilon^2(n)}{2(32n+1)} \cdot \frac{|Good_x|}{2^n} \cdot \frac{|Good_{r_3}|}{2^n} \cdot \frac{|Good_{r_2}|}{2^n}$$

$$\geq \frac{\varepsilon^5(n)}{128(32n+1)} \tag{A26}$$

where we can view r_3 and r_2 as the internal randomness of the inversion algorithm $\mathcal{A}'$. It is clear that if $\varepsilon(n)$ is non-negligible, it means that there exists polynomial $p(n)$ such that $\varepsilon(n) = 1/p(n)$, and then

$$\Pr_{\substack{x\leftarrow\{0,1\}^n \\ r_3\leftarrow\{0,1\}^n \\ r_2\leftarrow\{0,1\}^n}}[\mathcal{A}'(f(x),1^n) = \widetilde{x}] \geq \frac{1}{128(32n+1)p(n)^5} \tag{A27}$$

which as explained in Step 2 breaks second preimage resistance, Equation (A16). Since all the terms given in Step 3 ($\widetilde{B}_i, \widetilde{B}_i \oplus \widetilde{B}_j, \widetilde{B}_1 \oplus \widetilde{B}_2 \oplus \widetilde{B}_3$) are of the form $E(x, r_1, r_2, r_3)$ as in Equation (A18), our analysis suffices to prove that $\widetilde{B}_1\widetilde{B}_2\widetilde{B}_3$ is a hard-core function for f. □

Appendix C. Proof of Theorem 9

Lemma A2 (two-regular)**.** *If $\mathcal{G}$ is a family of bijective functions, then $\mathcal{F}$ is a family of two-regular functions.*

Proof. For every $y \in \text{Im}\, f_{k'} \subseteq R$, where $k' = (k_1, k_2)$:

1. Since $\text{Im}\, f_{k'} = \text{Im}\, g_{k_1}$ and g_{k_1} is bijective, there exist unique $x_1 := g_{k_1}^{-1}(y)$ such that $f_{k'}(x, 0) = g_{k_1}(x) = y$.
2. Since $\text{Im}\, f_{k'} = \text{Im}\, g_{k_2}$ and g_{k_2} is bijective, there exist unique $x_2 := g_{k_2}^{-1}(y)$ such that $f_{k'}(x, 1) = g_{k_2}(x) = y$.

Therefore, we conclude that:

$$\forall\, y \ \in \text{Im}\, f_{k'} :\ f_{k'}^{-1}(y) := \{(g_{k_1}^{-1}(y), 0), (g_{k_2}^{-1}(y), 1)\} \tag{A28}$$

□

Lemma A3 (trapdoor)**.** *If $\mathcal{G}$ is a family of bijective trapdoor functions, then $\mathcal{F}$ is a family of trapdoor functions.*

Proof. Let $y \in \text{Im}\, f_{k'} \subseteq R$. We construct the following inversion algorithm:

$Inv_{\mathcal{F}}(k', y, t'_k)$ □

1 : // $t'_k = (t_{k_1}, t_{k_2}), k' = (k_1, k_2)$
2 : $x_1 := Inv_{\mathcal{G}}(k_1, y, t_{k_1})$
3 : $x_2 := Inv_{\mathcal{G}}(k_2, y, t_{k_2})$
4 : **return** $(x_1, 0)$ and $(x_2, 1)$

Lemma A4 (one-way)**.** *If $\mathcal{G}$ is a family of bijective, one-way functions, then $\mathcal{F}$ is a family of one-way functions.*

Proof. We prove it by contradiction. We assume that a QPT adversary $\mathcal{A}$ can invert a function in $\mathcal{F}$ with non-negligible probability P (i.e., given $y \in \text{Im}\, f_{k'}$ to return a correct preimage of the form (x', b) with probability P). We then construct a QPT adversary $\mathcal{A}'$ that inverts any function in $\mathcal{G}$ with the same non-negligible probability P reaching the contradiction since $\mathcal{G}$ is one-way by assumption.

From Equation (A28), we know the two preimages of y are: (i) $(g_{k_1}^{-1}(y), 0)$ and (ii) $(g_{k_2}^{-1}(y), 1)$. We now construct an adversary $\mathcal{A}'$ that for $g_k : D \to R$, inverts any image $y = g_k(x)$ with the probability $P/2$.

$\mathcal{A}'(k_c, y)$

$r \leftarrow_\$ \{0,1\}$
$k_r := k_c$
$(k_{r'}, t_{k_{r'}}) \leftarrow_\$ Gen_{\mathcal{G}}(1^n)$
if $r == 0$ **then**
  $k' := (k_r, k_{r'})$
else
  $k' := (k_{r'}, k_r)$
$(x', b) \leftarrow \mathcal{A}(k', y)$
if $((b == r) \wedge (g_{k_r}(x') == y)$ **then**
  // $\mathcal{A}$ returns correct preimage that also corresponds to the challenge of $\mathcal{A}'$
  return x'
else // $\mathcal{A}$ failed in giving any of the preimages (happens with probability $1 - P$)
  // or the preimage returned corresponds to the r' that is not the challenge (happens with probability $P/2$)
  return 0

The inversion algorithm succeeds with $1 - ((1 - P) + P/2) = P/2$ and thus reaches a contradiction. □

Lemma A5 (second preimage resistance)**.** *If $\mathcal{G}$ is a family of bijective, one-way functions, then, any function $f \in \mathcal{F}$ is second preimage resistant.*

Proof. Assume there exists a PPT adversary $\mathcal{B}$ that given $k' = (k_1, k_2)$ and $(y, (x, b))$ such that $f_{k'}(x, b) = y$ can find (x', b') such that $f_{k'}(x', b') = y$ with non-negligible probability P. From Equation (A28), we know that the two preimages have different bs. We now construct a PPT adversary $\mathcal{B}'$ that inverts the function g_{k_c} with the same probability P, reaching a contradiction:

$\mathcal{B}'(k_c, y)$

$(k_2, t_{k_2}) \leftarrow_\$ Gen_{\mathcal{G}}(1^n)$
$x_2 \leftarrow g_{k_2}^{-1}(y)$// using the trapdoor t_{k_2}
$k' := (k_c, k_2)$
$(x, 0) \leftarrow \mathcal{B}(k', y, (x_2, 1))$// where y is an element from the image of $f_{k'}$
if $f_{k'}(x, 0) == f_{k'}(x_2, 1) == y$
return x
else // $\mathcal{B}$ failed to find a second preimage; happens with probability $(1 - P)$
return 0

□

Lemma A6 (quantum-safe)**.** *If $\mathcal{G}$ is a family of quantum-safe trapdoor functions, with properties as above, then $\mathcal{F}$ is also a family of quantum-safe trapdoor functions.*

Proof. The properties that require to be quantum-safe is one-wayness and second preimage resistance. Both these properties of $\mathcal{F}$ that we derived above were proved using reduction to the hardness (one-wayness) of $\mathcal{G}$. Therefore, if $\mathcal{G}$ is quantum-safe, its one-wayness is also quantum-safe and thus both properties of $\mathcal{F}$ are also quantum-safe. □

Appendix D. Proof of Theorem 11

In the following, we will denote by $f(s, e, c)$, the function $\texttt{REG2.Eval}_{\mathcal{P}}(k, (s, e, c))$ for k the index function obtained by $\texttt{REG2.Gen}_{\mathcal{P}}(1^n)$, and by s_0, e_0 the trapdoor information associated with this function f.

We now prove separately the δ-2 regularity, collision resistance, one-wayness and trapdoor property of the function in Definition 13.

Appendix D.1. δ-2 Regularity

Here, we describe how to achieve δ-2 regularity using the construction `FromInj` and specifically, the function in Definition 13.

This reduces to ensuring that the two function inputs (s, e) and $(s - s_0, e - e_0)$ both lie within the domain of the function. The input (s, e) is the result of the inversion algorithm, so it is by definition inside the domain. Additionally, as the first element of the domain is only required to be in $\mathbb{Z}_q^n$ and as $\mathbb{Z}_q$ is closed with subtraction mod q, then $s - s_0 \in \mathbb{Z}_q^n$ for any $s, s_0 \in \mathbb{Z}_q^n$. On the other hand, the second element of the domain is required to be in $\mathbb{Z}^m$, such that each component is bounded in absolute value by some value μ. In this case, we are not guaranteed that adding or subtracting two such elements the result is still in the domain. What we want to ensure is that with (at least) constant probability over the choice of (s, e) and (s_0, e_0), the result $(s - s_0, e - e_0)$ is in the domain of the function.

It is not difficult to show that if (s_0, e_0) is chosen arbitrarily from the domain of the function, then $(s - s_0, e - e_0)$ lies within the domain of the function only with inverse exponential in m probability. This is why we consider restricting e_0 to be within a subset of the domain. By suitable choice of this subset we can make the success probability (of having two preimages)—seen as a function in m—to be at least a constant value. Firstly, we remark that the exact probability of success can be explicitly computed. Indeed, if the trapdoor noise e_0 is sampled from a Gaussian of dimension m, and standard deviation σ, and if the noise e_1 is sampled uniformly from an hypercube C of length 2μ (both distribution being centred on 0) then the probability that $e_0 + e_1$ is still inside C is:

$$\left(\operatorname{erf}\left(\frac{\sqrt{2}\mu}{\sigma}\right) - \frac{\sigma}{\sqrt{2\pi}\mu}\left(1 - \exp\left(-2\left(\frac{\mu}{\sigma}\right)^2\right)\right)\right)^m$$

However, for simplicity, and because we do not aim to find optimal parameters, we will use a (simpler) lower bound of this probability (that will be less efficient by a factor of $\sqrt{m}$). To do that, remark that using in [46], we have that if $e_0 \in \mathbb{Z}^m$, such that each component of e_0 is sampled from a Gaussian distribution with parameter $\alpha' q$, then we have that every component of the vector e_0 is less than $\mu' := \alpha' q\sqrt{m}$ with overwhelming probability as m increases. Therefore, one can remark that, up to a negligible term, the Gaussian distribution with parameter $\alpha' q$ is "closer to 0" than the uniform distribution on $[-\alpha' q\sqrt{m}; \alpha' q\sqrt{m}]$ for sufficiently large m (i.e., for any x, the integral between $-x$ and x of the Gaussian distribution is bigger, up to a negligible term, than the integral of the uniform distribution). Therefore, to obtain a lower bound on the probability of having two preimages, we can consider that e_0 is sampled according to the uniform distribution on a hypercube of length $2\alpha' q\sqrt{m}$ rather than according to the Gaussian distribution of parameter $\alpha' q$. This simplifies our analysis, and allows us to find the subset in which e_0 must reside, as seen in the following lemma. Note also that if one does not want to do any assumption on the input distribution, and only assume that the infinity norm is smaller than μ', then the same lemma applies with the constant 4 replaced by 2.

Lemma A7 (Domain Addition)**.** *Let $V = \mathbb{R}^m$ be a vector space of dimension m, and let $\mathcal{D}_{m,\mu}$ be the uniform distribution inside the hypercube of dimension m and length 2μ centred on 0. Then, for any $\mu' < \mu$, we have:*

$$P_{m,\mu,\mu'} := \Pr\left[||e_0 + e_1||_\infty \leq \mu \mid e_0 \leftarrow \mathcal{D}_{m,\mu'}, e_1 \leftarrow \mathcal{D}_{m,\mu}\right] = \left(1 - \frac{\mu'}{4\mu}\right)^m$$

Moreover, if $\mu' = O(\frac{\mu}{m})$ then the probability $P_{m,\mu,\mu'}$ becomes lower bounded by a positive constant.

Proof. As $||e_0 + e_1||_\infty$ must be less than μ, which means that each component of the sum vector must be less than μ, and as each component of the 2 vectors e_0 and e_1 was independently sampled, then we can simplify our proof by considering that e_0 and e_1 are vectors in $\mathbb{R}$, essentially determining $P_{1,\mu,\mu'}$ and then, we can compute $P_{m,\mu,\mu'} = {P_{1,\mu,\mu'}}^m$.

Then, let us denote by E_1 the random variable sampled uniformly from $[-\mu, \mu]$, E_0 the random variable sampled uniformly from $[-\mu', \mu']$ and E the random variable obtained as $E = E_1 + E_0$. Therefore, $P_{1,\mu,\mu'} = Pr[-\mu \leq E \leq \mu]$.

Now, we can compute the density function of E using convolution:

$$f_E(e) = \int_{-\infty}^{\infty} f_{E_1}(e_1) \cdot f_{E_0}(e - e_1)\, de_1$$

where f_{E_1} and f_{E_0} are the probability density functions of E_1 and E_0 ($f_{E_1}(e_1) = \frac{1}{2\mu}$, when $e_1 \in [-\mu, \mu]$ and 0 elsewhere and $f_{E_0}(e_0) = \frac{1}{2\mu'}$, when $e_0 \in [-\mu', \mu']$ and 0 elsewhere).

Then, we are only interested in the cases when both the values of $f_{E_1}(e_1)$ and $f_{E_0}(e - e_1)$ are non-zero and for this we need to consider 3 cases for e, given by the intervals: $e \in [-\mu - \mu', \mu' - \mu] \bigcup [\mu' - \mu, \mu - \mu'] \bigcup [\mu - \mu', \mu + \mu']$. Thus, we can derive:

$$f_E(e) = \begin{cases} \int_{-\mu}^{e+\mu'} \frac{1}{4\mu\mu'}\, de_1\ , e \in [-\mu - \mu', \mu' - \mu] \\ \int_{e-\mu'}^{e+\mu'} \frac{1}{4\mu\mu'}\, de_1\ , e \in [\mu' - \mu, \mu - \mu'] \\ \int_{e-\mu'}^{\mu} \frac{1}{4\mu\mu'}\, de_1\ , e \in [\mu - \mu', \mu + \mu'] \end{cases}$$

Finally, we have that $Pr[-\mu \leq E \leq \mu] = \int_{-\mu}^{\mu} f_E(e)\, de = \int_{-\mu}^{-\mu+\mu'} f_E(e)\, de + \int_{-\mu+\mu'}^{\mu-\mu'} f_E(e)\, de + \int_{\mu-\mu'}^{\mu} f_E(e)\, de = \int_{-\mu}^{-\mu+\mu'} \frac{e+\mu'+\mu}{4\mu\mu'}\, de + \int_{-\mu+\mu'}^{\mu-\mu'} \frac{1}{2\mu}\, de + \int_{\mu-\mu'}^{\mu} \frac{\mu+\mu'-e}{4\mu\mu'}\, de = 1 - \frac{\mu'}{4\mu}$.

Consequently, we have $P_{m,\mu,\mu'} = (1 - \frac{\mu'}{4\mu})^m$.

Now, given that μ is a function of m, $\mu = \mu(m)$, we want to determine the values of μ', such that this probability (seen as a function in m) is at least a positive constant number.

- If $\lim_{m\to\infty} \frac{\mu'}{\mu} = 0$, then:

$$\lim_{m\to\infty} \left(1 - \frac{\mu'}{4\mu}\right)^m = \lim_{m\to\infty} \left(1 - \frac{\mu'}{4\mu}\right)^{\frac{4\mu}{\mu'}\frac{\mu' m}{4\mu}} = \left(\frac{1}{e}\right)^{\lim_{m\to\infty}\frac{\mu' m}{4\mu}}$$

  Now, what we require is that $\lim_{m\to\infty} \frac{\mu' m}{4\mu} = c \geq 0$, where c is a constant, as then, we have that the probability of success is at least a constant $\geq \left(\frac{1}{e}\right)^c$.

- If $\lim_{m\to\infty} \frac{\mu'}{\mu} > 0$ (and less than 1, as $0 < \mu' < \mu$), then:

$$\lim_{m\to\infty} \left(1 - \frac{\mu'}{4\mu}\right)^m = 0$$

Consequently, it is clear that in order to get a positive constant lower bound for the success probability, we must have:

$$\mu' = c \cdot \frac{4\mu}{m},\ \ c \geq 0$$

□

Thus, in our case, if e_1 is sampled uniformly on a hypercube of length 2μ and e_0 from a Gaussian with parameter $\alpha' q$, by replacing the actual values of $\mu = \alpha q\sqrt{m}$ and $\mu' := \alpha' q\sqrt{m}$, what we require is that:

$$\alpha' = c \cdot \frac{4\alpha}{m}, \quad c \geq 0$$

Appendix D.2. Collision Resistance

We start by the observation that for the choices of Definition 13, no QPT adversary can infer the trapdoor information (s_0, e_0), as determining s_0 from $k = (A, b_0)$ would be equivalent to solving $\mathrm{LWE}_{q,\bar{\Psi}_{\alpha' q}}$:

Corollary A1 (One-wayness of the trapdoor ([46], Theorem 1.1))**.** *Under the* SIVP_γ *(with* $\gamma = \mathrm{poly}(n)$*) assumption, no QPT adversary can recover the trapdoor information* (s_0, e_0)*.*

Lemma A8 (Collision resistance)**.** *The function* f *defined in Definition 13 is collision resistant if the parameters are chosen accordingly to Theorem 11 assuming that* SIVP_γ *is hard.*

Proof. By contradiction, let us suppose that this function is not collision resistant. Then, there exist two pairs (s_1, e_1), (s_2, e_2) such that $f(s_1, e_1, 0) = y = f(s_2, e_2, 1)$. Note that the last bits are necessary different since the two functions that fix the last bit, are injective when the error is smaller than r_{max} (according to ([44], Theorem 5.4)). By the definition of f, $||e_1||_\infty \leq \mu$ and $||e_2||_\infty \leq \mu$, i.e., both e_1 and e_2 has Euclidean norm smaller than $\sqrt{m}\mu$. Then, by definition, $y = f(s_2, e_2, 1) = f(s_2, e_2, 0) + f(s_0, e_0, 0) = A(s_2 + s_0) + (e_2 + e_0)$. Now, we remark that with overwhelming probability (over the choice of the trapdoor), $||e_0||_2 \leq \mu'\sqrt{m}$ as stated in ([46], Lemma 2.5), so in this case, $||e_2 + e_0||_2 \leq \sqrt{m}(\mu + \mu') \leq r_{max}$ (last assumption of Theorem 11). Then, according to ([44], Theorem 5.4), there is exactly one element (s, e) with e of length smaller than r_{max} such that $As + e = y$. As a result that (s_1, e_1) is a solution, we then have that: $s_2 + s_0 = s_1$ and $e_2 + e_0 = e_1$, i.e., $e_0 = e_1 - e_2$ and $s_0 = s_1 - s_2 \mod q$. Hence, it is possible to deduce the trapdoor information s_0 and e_0 from the collision pair, which is impossible by Corollary A1. □

Appendix D.3. One-Wayness

One could imagine that the one-wayness of the resulting function of Definition 13 is implied by the one-wayness of the function in [44] (as is the case in Lemma 4). However, we need more care here, since in our construction the error term e is not sampled from a Gaussian distribution with suitable parameters (unlike the error term e_0). We remark that while other ways to prove the one-wayness are possible, we give here one proof that uses the previous two lemmata.

Lemma A9 (Collision resistance to one-wayness)**.** *Let* $f : A \to B \cup \perp$ *(with* $\perp \notin B$*), where* A *is finite and can be efficiently sampled uniformly and let* C *be the set of all* $y \in B$ *that admit 2 preimages. If the restriction of* f *to the set* $f^{-1}(B)$ *is a collision resistant function that admits with non-negligible probability two preimages for any* y *from its image and if* $\frac{|f^{-1}(C)|}{|A|}$ *is non-negligible, then* f *restricted to the set* $f^{-1}(C)$ *is a one-way function.*

Proof. By contradiction: suppose that f is not one-way on C, i.e., with a non-negligible probability we can find a preimage of y for y uniformly sampled in C, and from this we can show how to find a collision. The idea is to sample an input $x \in A$, and then compute $y := f(x)$. Then, as $\frac{|f^{-1}(C)|}{|A|}$ is non-negligible, we know that with non-negligible probability this y will have two preimages. Now, with non-negligible probability, this function will be easy to invert and one gets x'. As a result that we sample uniformly at the step before, we have the same probability to sample one image or the other, so with probability $1/2$, $x' \neq x$, therefore, we found a collision. □

Corollary A2 (One-wayness from Lemmas A8 and A9)**.** *The function defined in Definition 13 is one-way for all y that admit two preimages, under the* SIVP_γ *hardness assumption, when the parameters are chosen accordingly to Theorem 11.*

Appendix D.4. Trapdoor

We want to prove that using the trapdoor information of the `REG2` construction, which consists of (s_0, e_0) and t_k, the trapdoor information of the `LWE` function, we can efficiently derive the preimages of an output b of `REG2.Eval`. Firstly, we notice that to find all the preimages, we can simply run `LWE.Inv` on b as well as on $b - b_0$ and if we succeed we take only the preimages that lie in the input domain, i.e., whose error part e is bounded in infinity norm by μ: $||e||_\infty \leq \mu$. As a result that the function is injective, these are all the possible preimages. However, because we are interested only in the case when there are exactly two preimages, the function `REG2.Inv` can also do the following: we first run `LWE.Inv` on b and obtain (s_1, e_1). Then, the inversion is completed by returning $(s_1, e_1, 0)$ and $(s_1 - s_0, e_1 - e_0, 1)$, which are both valid preimages, if and only if the function has two preimages (see Lemma A8 for more details).

Appendix E. Proof of Lemma 7

Proof. Using the following explicit values for the parameters of the Micciancio and Peikert injective trapdoor function [44], we want to prove that they fulfil all of the requirements of Theorem 11:

$$\begin{aligned} n &= \lambda \\ k &= 5\lceil \log(n) \rceil + 21 \\ q &= 2^k \\ \bar{m} &= 2n \\ \omega &= nk \\ m &= \bar{m} + \omega \\ \mu &= \left\lceil 2mn\sqrt{2+k} \right\rceil \\ \mu' &= \mu / m \\ B &= 2 \end{aligned}$$

and α, α', C are defined as in Theorem 11. Now, let us proof that these parameters satisfy all the requirements.

- The first three requirements are trivially satisfied.
- In the forth condition, the only difficulty is to show that $\alpha < 1$. By definition,

$$\begin{aligned} \alpha = \frac{m\mu}{\sqrt{m}mq} = \frac{\mu}{\sqrt{m}q} &= \frac{\left\lceil 2mn\sqrt{2+k} \right\rceil}{\sqrt{m}q} \leq \frac{4mn\sqrt{2+k}}{\sqrt{m}q} \leq \frac{8mn\sqrt{k}}{\sqrt{m}q} \\ \leq \frac{8\sqrt{m}nk}{q} \leq \frac{8\sqrt{2n+nk}nk}{2^{21}n^5} &\leq \frac{8\sqrt{2nk}nk}{2^{21}n^5} \leq \frac{16(nk)^{3/2}}{2^{21}n^5} \\ \leq \frac{16(n(5(\log(n)+1)+21))^{3/2}}{2^{21}n^5} &\leq \frac{16(5\times 21n^2)^{3/2}}{2^{21}n^5} \leq \frac{16\times 1076n^3}{2^{21}n^5} < \frac{1}{n^2} \leq 1 \end{aligned}$$

- Now, let us show the fifth condition, i.e., $\alpha' q \geq 2\sqrt{n}$. First we note that $\alpha' q := \frac{\mu}{\sqrt{m}m} \geq 2\sqrt{n} \Leftrightarrow \mu \geq 2\sqrt{n}m\sqrt{m} = 2mn\sqrt{2+k}$. Then, by defining $\mu = \left\lceil 2mn\sqrt{2+k} \right\rceil$, the condition is satisfied.
- For the fifth condition, i.e., $\frac{n}{\alpha'}$ is $\mathrm{poly}(n)$, we just need to remark that $1/\alpha' = \frac{m^{3/2}q}{\mu} < m^{3/2}q$, and that both m and q are $\mathrm{poly}(n)$.

- Finally, to show that the last condition is satisfied, we note that:

$$\sqrt{m}\mu < \frac{q}{2B\sqrt{\left(C\cdot(\alpha\cdot q)\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1}} - \mu'\sqrt{m} \tag{A29}$$

$$= \frac{q}{4\sqrt{\left(C\cdot\frac{\mu}{\sqrt{m}}\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1}} - \frac{\mu}{\sqrt{m}} \tag{A30}$$

if and only if

$$A := 4\left(\sqrt{m}+\frac{1}{\sqrt{m}}\right)\mu\sqrt{\left(C\cdot\frac{\mu}{\sqrt{m}}\cdot(\sqrt{2n}+\sqrt{kn}+\sqrt{n})\right)^2+1} \leq q$$

Now, let us suppose that $k := u\lceil\log(n)\rceil + v$ with $u \leq 5$ and $v \geq 19$ and we need to find u, v such that $A \leq 2^k$. Note that we will include v in some constants and then find the good v at the end. First, remark that:

$$\sqrt{m}+\frac{1}{\sqrt{m}} = \sqrt{m}(1+\frac{1}{m}) \tag{A31}$$

$$= \sqrt{m}(1+\frac{1}{n(2+k)}) \tag{A32}$$

$$\leq \sqrt{m}(1+\frac{1}{2+k}) \tag{A33}$$

$$\leq \sqrt{m}\underbrace{(1+\frac{1}{2+v})}_{\gamma_0} = \gamma_0\sqrt{m} \tag{A34}$$

So now,

$$A \leq 4C\gamma_0\mu^2\sqrt{kn}\sqrt{\left(1+\sqrt{\frac{2}{k}}+\frac{1}{\sqrt{k}}\right)^2+\frac{1}{kn\left(C\cdot\frac{\mu}{\sqrt{m}}\right)^2}}$$

$$= 4C\gamma_0\left\lceil 2mn\sqrt{2+k}\right\rceil^2\sqrt{kn}\sqrt{\left(1+\sqrt{\frac{2}{k}}+\frac{1}{\sqrt{k}}\right)^2+\frac{1}{kn\left(C\cdot\frac{\lceil 2mn\sqrt{2+k}\rceil}{\sqrt{m}}\right)^2}}$$

$$\leq 4C\gamma_0\left\lceil 2mn\sqrt{2+k}\right\rceil^2\sqrt{kn}\underbrace{\sqrt{\left(1+\sqrt{\frac{2}{v}}+\frac{1}{\sqrt{v}}\right)^2+\frac{1}{v(2C\sqrt{2+v})^2}}}_{\gamma_1}$$

$$\leq 4C\gamma_0\gamma_1\left(2mn\sqrt{2+k}+1\right)^2\sqrt{kn}$$

$$= 4C\gamma_0\gamma_1\left(2n^2(2+k)^{3/2}+1\right)^2\sqrt{kn}$$

$$= 16C\gamma_0\gamma_1n^4(2+k)^3\left(1+\frac{1}{2n^2(2+k)^{3/2}}\right)^2\sqrt{kn}$$

$$\leq 16C\gamma_0\gamma_1n^4(2+k)^3\underbrace{\left(1+\frac{1}{2(2+v)^{3/2}}\right)^2}_{\gamma_2}\sqrt{kn}$$

$$\leq 16C\gamma_0\gamma_1\gamma_2n^4\left(k\left(1+\frac{2}{k}\right)\right)^3\sqrt{kn}$$

$$
\begin{aligned}
&\leq 16C\gamma_0\gamma_1\gamma_2 n^4 k^3 \underbrace{\left(1+\frac{2}{v}\right)^3}_{\gamma_3} \sqrt{kn} \\
&\leq 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2}(u\lceil\log(n)\rceil + v)^{7/2} \\
&= 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2} v^{7/2}\left(1+\frac{u\lceil\log(n)\rceil}{v}\right)^{7/2} \\
&\leq 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2} v^{7/2}\left(1+\frac{5\lceil\log(n)\rceil}{19}\right)^{7/2} \\
&\leq 16C\gamma_0\gamma_1\gamma_2\gamma_3 n^{9/2} v^{7/2} 3n^{1/2} \\
&\leq 48C\gamma_0\gamma_1\gamma_2\gamma_3 v^{7/2} n^5
\end{aligned}
$$

Finally, we observe that if $v = 21$ and $u = 5$, we have $A \leq 2^{v+u\lceil\log(n)\rceil} = 2^k$, which concludes the proof.

□

References

1. Elkouss, D.; Lipinska, V.; Goodenough, K.; Rozpedek, F.; Kalb, N.; van Dam, S.; Le Phuc, T.; Murta, G.; Humphreys, P.; Taminiau, T.; et al. Quantum internet: The certifiable road ahead. In Proceedings of the APS Meeting Abstracts, New Orleans, LA, USA, 13–17 March 2017.
2. Broadbent, A.; Schaffner, C. Quantum cryptography beyond quantum key distribution. *Des. Codes Cryptogr.* **2016**, *78*, 351–382. [CrossRef]
3. Fitzsimons, J.F. Private quantum computation: An introduction to blind quantum computing and related protocols. *Npj Quantum Inf.* **2017**, *3*, 23. [CrossRef]
4. Broadbent, A.; Jeffery, S. Quantum homomorphic encryption for circuits of low T-gate complexity. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 609–629.
5. Dulek, Y.; Schaffner, C.; Speelman, F. Quantum homomorphic encryption for polynomial-sized circuits. In *Annual Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–32.
6. Alagic, G.; Dulek, Y.; Schaffner, C.; Speelman, F. Quantum fully homomorphic encryption with verification. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 438–467.
7. Liang, M. Quantum fully homomorphic encryption scheme based on universal quantum circuit. *Quantum Inf. Process.* **2015**, *14*, 2749–2759. [CrossRef]
8. Ouyang, Y.; Tan, S.H.; Fitzsimons, J. Quantum homomorphic encryption from quantum codes. *arXiv* **2015**, arXiv:1508.00938.
9. Tan, S.H.; Kettlewell, J.A.; Ouyang, Y.; Chen, L.; Fitzsimons, J.F. A quantum approach to homomorphic encryption. *Sci. Rep.* **2016**, *6*, 33467. [CrossRef] [PubMed]
10. Lai, C.Y.; Chung, K.M. On statistically-secure quantum homomorphic encryption. *arXiv* **2017**, arXiv:1705.00139.
11. Mantri, A.; Pérez-Delgado, C.A.; Fitzsimons, J.F. Optimal blind quantum computation. *Phys. Rev. Lett.* **2013**, *111*, 230502. [CrossRef]
12. Giovannetti, V.; Maccone, L.; Morimae, T.; Rudolph, T.G. Efficient universal blind quantum computation. *Phys. Rev. Lett.* **2013**, *111*, 230501. [CrossRef]
13. Armknecht, F.; Gagliardoni, T.; Katzenbeisser, S.; Peter, A. General impossibility of group homomorphic encryption in the quantum world. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 556–573.
14. Yu, L.; Pérez-Delgado, C.A.; Fitzsimons, J.F. Limitations on information-theoretically-secure quantum homomorphic encryption. *Phys. Rev. A* **2014**, *90*, 050303. [CrossRef]
15. Aaronson, S.; Cojocaru, A.; Gheorghiu, A.; Kashefi, E. Complexity-Theoretic Limitations on Blind Delegated Quantum Computation. In Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), Patras, Greece, 8–12 July 2019.
16. Newman, M.; Shi, Y. Limitations on Transversal Computation through Quantum Homomorphic Encryption. *arXiv* **2017**, arXiv:1704.07798.
17. Mantri, A.; Demarie, T.F.; Menicucci, N.C.; Fitzsimons, J.F. Flow ambiguity: A path towards classically driven blind quantum computation. *Phys. Rev. X* **2017**, *7*, 031004. [CrossRef]
18. Mahadev, U. Classical Homomorphic Encryption for Quantum Circuits. In Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2018), Paris, France, 7–9 October 2018; Thorup, M., Ed.; IEEE Computer Society: Washington, DC, USA, 2018; pp. 332–338.

19. Brakerski, Z. Quantum FHE (Almost) As Secure As Classical. In *Advances in Cryptology—CRYPTO 2018*; Springer International Publishing: Cham, Switzerland, 2018; pp. 67–95.
20. Badertscher, C.; Cojocaru, A.; Colisson, L.; Kashefi, E.; Leichtle, D.; Mantri, A.; Wallden, P. Security Limitations of Classical-Client Delegated Quantum Computing. In *Advances in Cryptology—ASIACRYPT 2020*; Springer: Berlin/Heidelberg, Germany, 2020._23. [CrossRef]
21. Broadbent, A.; Fitzsimons, J.; Kashefi, E. Universal blind quantum computation. In Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS '09), Atlanta, GA, USA, 25–27 October 2009; IEEE Computer Society: Washington, DC, USA, 2009; pp. 517–526. [CrossRef]
22. Cojocaru, A.; Colisson, L.; Kashefi, E.; Wallden, P. On the possibility of classical client blind quantum computing. *arXiv* **2018**, arXiv:1802.08759.
23. Cojocaru, A.; Colisson, L.; Kashefi, E.; Wallden, P. QFactory: Classically-Instructed Remote Secret Qubits Preparation. In *Advances in Cryptology—ASIACRYPT 2019*; Galbraith, S.D., Moriai, S., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2019, pp. 615–645.
24. Gheorghiu, A.; Vidick, T. Computationally-Secure and Composable Remote State Preparation. In Proceedings of the 2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS), Baltimore, MA, USA, 9–12 November 2019; pp. 1024–1033.
25. Zhang, J. Succinct Blind Quantum Computation Using a Random Oracle. *arXiv* **2020**, arXiv:2004.12621.
26. Pirandola, S.; Andersen, U.L.; Banchi, L.; Berta, M.; Bunandar, D.; Colbeck, R.; Englund, D.; Gehring, T.; Lupo, C.; Ottaviani, C.; et al. Advances in quantum cryptography. *Adv. Opt. Photon.* **2020**, *12*, 1012–1236. [CrossRef]
27. Wallden, P.; Kashefi, E. Cyber Security in the Quantum Era. *Commun. ACM* **2019**, *62*, 120. [CrossRef]
28. Brakerski, Z.; Christiano, P.; Mahadev, U.; Vazirani, U.; Vidick, T. Certifiable Randomness from a Single Quantum Device. *arXiv* **2018**, arXiv:1804.00640.
29. Freeman, D.M.; Goldreich, O.; Kiltz, E.; Rosen, A.; Segev, G. More constructions of lossy and correlation-secure trapdoor functions. In *International Workshop on Public Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 279–295.
30. Fitzsimons, J.F.; Kashefi, E. Unconditionally verifiable blind computation. *arXiv* **2012**, arXiv:1203.5217.
31. Broadbent, A. How to Verify a Quantum Computation. *arXiv* **2015**, arXiv:1509.09180.
32. Ferracin, S.; Kapourniotis, T.; Datta, A. Towards minimising resources for verification of quantum computations. *arXiv* **2017**, arXiv:1709.10050.
33. Bennett, C.H.; Brassard, G. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* **2014**, *560*, 7–11. [CrossRef]
34. Bozzio, M.; Orieux, A.; Vidarte, L.T.; Zaquine, I.; Kerenidis, I.; Diamanti, E. Experimental investigation of practical unforgeable quantum money. *Npj Quantum Inf.* **2018**, *4*, 5. [CrossRef]
35. Pappa, A.; Chailloux, A.; Diamanti, E.; Kerenidis, I. Practical quantum coin flipping. *Phys. Rev. A* **2011**, *84*, 052305. [CrossRef]
36. Wallden, P.; Dunjko, V.; Kent, A.; Andersson, E. Quantum digital signatures with quantum-key-distribution components. *Phys. Rev. A* **2015**, *91*, 042304. [CrossRef]
37. Kashefi, E.; Wallden, P. Garbled Quantum Computation. *Cryptography* **2017**, *1*, 6. [CrossRef]
38. Kashefi, E.; Music, L.; Wallden, P. The Quantum Cut-and-Choose Technique and Quantum Two-Party Computation. *arXiv* **2017**, arXiv:1703.03754.
39. Kashefi, E.; Pappa, A. Multiparty Delegated Quantum Computing. *Cryptography* **2017**, *1*, 12. [CrossRef]
40. Broadbent, A.; Gutoski, G.; Stebila, D. Quantum One-Time Programs. In *Advances in Cryptology—CRYPTO 2013*; Canetti, R. Garay, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8043, pp. 344–360._20. [CrossRef]
41. Ciampi, M.; Cojocaru, A.; Kashefi, E.; Mantri, A. Secure Quantum Two-Party Computation: Impossibility and Constructions. *arXiv* **2020**, arXiv:2010.07925.
42. Goldreich, O.; Levin, L.A. A Hard-core Predicate for All One-way Functions. In Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC '89), Washington, DC, USA, 15–17 May 1989; ACM: New York, NY, USA, 1989; pp. 25–32. [CrossRef]
43. Vazirani, U.V.; Vazirani, V.V. Efficient and Secure Pseudo-Random Number Generation (Extended Abstract). In *Advances in Cryptology*; Blakley, G.R., Chaum, D., Eds.; Springer: Berlin/Heidelberg, Germany, 1985; pp. 193–202.
44. Micciancio, D.; Peikert, C. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *Advances in Cryptology—EUROCRYPT 2012*; Pointcheval, D., Johansson, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 700–718.
45. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography*, 2nd ed.; Chapman & Hall/CRC: Boca Raton, FL, USA, 2014.
46. Regev, O. 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), Baltimore, MD, USA, 22–24 May 2005; ACM: New York, NY, USA, 2005; pp. 84–93. [CrossRef]
47. Peikert, C. 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), Bethesda, MD, USA, 31 May–2 June 2009; ACM: New York, NY, USA, 2009; pp. 333–342. [CrossRef]
48. Aaronson, S. Quantum computing, postselection, and probabilistic polynomial-time. *Proc. R. Soc. Lond. Ser. A* **2005**, *461*, 3473–3482. [CrossRef]

49. Greenberger, D.M.; Horne, M.A.; Zeilinger, A. Going beyond Bell's theorem. In *Bell's Theorem, Quantum Theory and Conceptions of the Universe*; Springer: Berlin/Heidelberg, Germany, 1989; pp. 69–72.
50. Vaikuntanathan, V. Advanced Topics in Cryptography: Lattices. Available online: https://people.csail.mit.edu/vinodv/6876-Fall2015/L13.pdf (accessed on 7 December 2018).