Secure Software Leasing

Prabhanjan Ananth[1(✉)] and Rolando L. La Placa[2]

[1] UC Santa Barbara, Santa Barbara, USA
prabhanjan@cs.ucsb.edu
[2] MIT, Santa Barbara, USA
rlaplaca@mit.edu

Abstract. Formulating cryptographic definitions to protect against software piracy is an important research direction that has not received much attention. Since natural definitions using classical cryptography are impossible to achieve (as classical programs can always be copied), this directs us towards using techniques from quantum computing. The seminal work of Aaronson [CCC'09] introduced the notion of quantum copy-protection precisely to address the problem of software anti-piracy. However, despite being one of the most important problems in quantum cryptography, there are no provably secure solutions of quantum copy-protection known for *any* class of functions.

We formulate an alternative definition for tackling software piracy, called secure software leasing (SSL). While weaker than quantum copy-protection, SSL is still meaningful and has interesting applications in software anti-piracy.

We present a construction of SSL for a subclass of evasive circuits (that includes natural implementations of point functions, conjunctions with wild cards, and affine testers) based on concrete cryptographic assumptions. Our construction is the first provably secure solution, based on concrete cryptographic assumptions, for software anti-piracy. To complement our positive result, we show, based on cryptographic assumptions, that there is a class of quantum unlearnable functions for which SSL does not exist. In particular, our impossibility result also rules out quantum copy-protection [Aaronson CCC'09] for an arbitrary class of quantum unlearnable functions; resolving an important open problem on the possibility of constructing copy-protection for arbitrary quantum unlearnable circuits.

1 Introduction

Almost all proprietary software requires a legal document, called software license, that governs the use against illegal distribution of software, also referred to as pirating. The main security requirement from such a license is that any malicious user no longer has access to the functionality of the software after the lease associated with the software license expires. While ad hoc solutions existed in the real world, for a long time, no theoretical treatment of this problem was known.

A. Canteaut and F.-X. Standaert (Eds.): EUROCRYPT 2021, LNCS 12697, pp. 501–530, 2021.
https://doi.org/10.1007/978-3-030-77886-6_17

This was until Aaronson, who in his seminal work [3] introduced and formalized the notion of quantum software copy-protection, a quantum cryptographic primitive that uses quantum no-cloning techniques to prevent pirating of software by modeling software as boolean functions. Roughly speaking, quantum copy-protection says[1] that given a quantum state computing a function f, the adversary cannot produce two quantum states (possibly entangled) such that each of the states individually computes f. This prevents a pirate from being able to create a new software from his own copy and re-distribute it; of course it can circulate its own copy to others but it will lose access to its own copy.

Need for Alternate Notions. While quantum copy-protection does provide a solution for software piracy, constructing quantum copy-protection has been notoriously difficult. Despite being introduced more than a decade ago, not much is known on the existence of quantum copy-protection. There are no known provably secure constructions of quantum copy-protection for *any* class of circuits. All the existing constructions of quantum copy-protection are either proven in an oracle model [3,5] or are heuristic[2] candidates for very simple functions such as point functions [3]. In a recent blog post, Aaronson [2] even mentioned constructing quantum copy-protection from cryptographic assumptions as one of the five big questions he wishes to solve.

This not only prompted us to explore the possibility of copy-protection but also look for alternate notions to protect against software piracy. Specifically, we look for application scenarios where the full power of quantum copy-protection is not needed and it suffices to settle for weaker notions. Let us consider one such example.

Example: Anti-Piracy Solutions for Microsoft Office. Microsoft Office is one of the most popular software tools used worldwide. Since Microsoft makes a sizeable portion of their revenue from this tool [1], it is natural to protect Microsoft Office from falling prey to software piracy. A desirable requirement is that pirated copies cannot be sold to other users such that these copies can run successfully on other Microsoft Windows systems. Importantly, it does not even matter if the pirated copies can be created as long as they cannot be executed on other Windows systems; this is because, only the pirated copies that run on Windows systems are the ones that bite into the revenue of Microsoft. Indeed, there are open source versions of Office publicly available but our aim is to prevent these open source versions from being sold off as authentic versions of Microsoft Office software.

This suggests that instead of quantum copy-protection – which prevents the adversary from creating *any* pirated copy of the copy-protected software – we can consider a weaker variant that only prevents the adversary from being able to create *authenticated* pirated copies (for instance, that runs only on specific

[1] More generally, Aaronson considers the setting where the adversary gets multiple copies computing f and not just one.

[2] That is, there is no known reduction to concrete cryptographic assumptions.

operating systems). To capture this, we present a new definition called secure software leasing.

Our Work: Secure Software Leasing (SSL). Roughly speaking, a secure leasing scheme allows for an authority (the lessor[3]) to lease a classical circuit C to a user (the lessee[4]) by providing a corresponding quantum state ρ_C. The user can execute ρ_C to compute C on any input it desires. Leases can expired, requiring ρ_C to be returned at a later point in time, specified by the lease agreement. After it returns the state, we require the security property that the lessee can no longer compute C.

In more detail, a secure software leasing scheme (SSL) for a family of circuits $\mathcal{C}$ is a collection, $(\mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$, of quantum polynomial-time algorithms (QPT) satisfying the following conditions. $\mathsf{Gen}(1^\lambda)$, on input a security parameter λ, outputs a secret key sk that will be used by a lessor to validate the states being returned after the expiration of the lease. For any circuit $C : \{0,1\}^n \to \{0,1\}^m$ in $\mathcal{C}$, $\mathsf{Lessor}(\mathsf{sk}, C)$ outputs a quantum state ρ_C, where ρ_C allows Run to evaluate C. Specifically, for any $x \in \{0,1\}^n$, we want that $\mathsf{Run}(\rho_C, x) = C(x)$; this algorithm is executed by the lessee. Finally, $\mathsf{Check}(\mathsf{sk}, \rho_C)$ checks if ρ_C is a valid leased state. Any state produced by the lessor is a valid state and will pass the verification check.

A SSL scheme can have two different security guarantees depending on whether the leased state is supposed to be returned or not.

- *Infinite-Term Lessor Security*: In this setting, there is no time duration associated with the leased state and hence, the user can keep this leased state forever[5]. Informally, we require the guarantee that the lessee, using the leased state, cannot produce two *authenticated* copies of the leased state. Formally speaking, any (malicious) QPT user $\mathcal{A}$ holding a leased state $\mathcal{A}(\rho_C)$ (produced using classical circuit C) cannot output a (possibly entangled) bipartite state σ^* such that both $\sigma_1^* = \mathrm{Tr}_2[\sigma^*]$ and $\sigma_2^* = \mathrm{Tr}_1[\sigma^*]$ can be used to compute C with Run.
- *Finite-Term Lessor Security*: On the other hand, we could also consider a weaker setting where the leased state is associated with a fixed term. In this setting, the lessee is obligated to return back the leased state after the term expires. We require the property that after the lessee returns back the state, it can no longer produce another *authenticated* state having the same functionality as the leased state.

  Formally speaking, we require that any (malicious) QPT user $\mathcal{A}$ holding a leased state ρ_C (produced using C) cannot output a (possibly entangled) bipartite states σ^* such that $\sigma_1^* := \mathrm{Tr}_2[\sigma^*]$[6] passes the lessor's verification $(\mathsf{Check}(\mathsf{sk}, \sigma_1^*) = 1)$ and such that the resulting state, after the first register

[3] The person who leases the software to another.
[4] The person to whom the software is being leased to.
[5] Although the lessor will technically be the owner of the leased state.
[6] This denotes tracing out the second register.

has been verified by the lessor, on the second register, σ_2^*, can also be used to evaluate C with the Run algorithm, $\mathsf{Run}(\sigma_2^*, x) = C(x)$.

A SSL scheme satisfying infinite-term security would potentially be useful to tackle the problem of developing anti-piracy solutions for Microsoft Office. However, there are scenarios where finite-term security suffices. We mention two examples below.

Trial Versions. Before releasing the full version of a program C, a software vendor might want to allow a selected group of people[7] to run a beta version of it, C_β, in order to test it and get user feedback. Naturally, the vendor would not want the beta versions to be pirated and distributed more widely. Again, they can lease the beta version C_β, expecting the users to return it back when the beta test is over. At this point, they would know if a user did not return their beta version and they can penalize such a user according to their lease agreement.

Subscription Models. Another example where finite-term SSL would be useful is for companies that use a subscription model for their revenue. For example, Microsoft has a large library of video games for their console, the Xbox, which anyone can have access to for a monthly subscription fee. A malicious user could subscribe in order to have access to the collection of games, then make copies of the games intending to keep them after cancelling the subscription. The same user will not be able to make another copy of a game that also runs on Xbox.

1.1 Our Results

We present a construction of SSL for a restricted class of unlearnable circuits; in particular, our construction is defined for a subclass of evasive circuits. This demonstrates the first provably secure construction for the problem of software anti-piracy in the standard model (i.e., without using any oracles).

Our construction does not address the possibility of constructing SSL for an arbitrary class of unlearnable circuits. Indeed, given the long history of unclonable quantum cryptographic primitives (see Section A) along with the recent boom in quantum cryptographic techniques [8,13,20,22–25,35,36,41], one might hope that existing techniques could lead us to achieve a general result. We show, rather surprisingly, assuming cryptographic assumptions, there exists a class of unlearnable circuits such that no SSL exists for this class. This also rules out the existence of quantum copy-protection for arbitrary unlearnable functions[8]; *thus resolving an important open problem in quantum cryptography.*

[7] For instance, they could be engineers assigned to test whether the beta version contains bugs.

[8] Both the notions (quantum copy-protection and secure software leasing) are only meaningful for unlearnable functions: if a function is learnable, then one could learn the function from the quantum state and create another authenticated quantum state computing the same function.

We explain our results in more detail. We first start with the negative result before moving on to the positive result.

Impossibility Result. To demonstrate our impossibility result, we identify a class of classical circuits $\mathcal{C}$ that we call a *de-quantumizable* circuit class. This class has the nice property that given *any* efficient quantum implementation of $C \in \mathcal{C}$, we can efficiently 'de-quantumize' it to obtain a classical circuit $C' \in \mathcal{C}$ that has the same functionality as C. If $\mathcal{C}$ is learnable then, from the definition of learnability, there could be a QPT algorithm that finds C'. To make the notion interesting and non-trivial, we add the additional requirement that this class of circuits is quantum unlearnable. A circuit class $\mathcal{C}$ is quantum unlearnable if given black-box access to $C \in \mathcal{C}$, any QPT algorithm cannot find a quantum implementation of C.

We show the existence of a de-quantumizable circuit class from cryptographic assumptions.

Proposition 1 (Informal). *Assuming the quantum hardness of learning with errors (QLWE), and asssuming the existence of quantum fully homomorphic encryption*[9] *(QFHE), there exists a de-quantumizable class of circuits.*

We show how non-black-box techniques introduced in seemingly different contexts – proving impossibility of obfuscation [7,12,19] and constructing zero-knowledge protocols [9,14,16] – are relevant to proving the above proposition. We give an overview, followed by a formal construction, in Sect. 3.

We then show that for certain de-quantumizable class of circuis, there does not exist a SSL scheme (with either finite or infinite-term security) for this class. Combining this with the above proposition, we have the following:

Theorem 1 (Informal). *Assuming the quantum hardness of learning with errors (QLWE), and asssuming the existence of quantum fully homomorphic encryption (QFHE), there exists a class of quantum unlearnable circuits $\mathcal{C}$ such that there is no SSL for $\mathcal{C}$.*

On the Assumption of QFHE: There are lattice-based constructions of QFHE proposed by [17,35] although we currently don't know how to base them solely on the assumption of LWE secure against QPT adversaries (QLWE). Brakerski [17] shows that the security of QFHE can be based on QLWE and a circular security assumption.

Impossibility of Quantum Copy-Protection. Since copy-protection implies SSL, we have the following result.

[9] We need additional properties from the quantum fully homomorphic encryption scheme but these properties are natural and satisfied by existing schemes [17,35]. Please refer to full version for a precise description of these properties.

Corollary 1 (Informal). *Assuming the quantum hardness of learning with errors (QLWE), and asssuming the existence of quantum fully homomorphic encryption (QFHE), there exists a class of quantum unlearnable circuits $\mathcal{C}$ that cannot be quantumly copy-protected.*

Main Construction. Our impossibility result does not rule out the possibility of constructing SSL schemes for specific circuit classes. For instance, it does not rule out the feasibility of SSL for *evasive functions*; this is a class of functions with the property that given black-box access, an efficient algorithm cannot find an accepting input (an input on which the output of the function is 1).

We identify a subclass of evasive circuits for which we can construct SSL. infinite

Searchable Compute-and-Compare Circuits. We consider the following circuit class $\mathcal{C}$: every circuit in $\mathcal{C}$, associated with a circuit C and a lock α, takes as input x and outputs 1 iff $C(x) = \alpha$. This circuit class has been studied in the cryptography literature in the context of constructing program obfuscation [32, 39]. We require this circuit class to additionally satisfy a *searchability* condition: there is an efficient (classical) algorithm, denoted by $\mathcal{S}$, such that given any $C \in \mathcal{C}$, $\mathcal{S}(C)$ outputs x such that $C(x) = 1$.

There are natural and interesting sub-classes of compute-and-compare circuits:

- Point circuits $C(\alpha, \cdot)$: the circuit $C(\alpha, \cdot)$ is a point circuit if it takes as input x and outputs $C(\alpha, x) = 1$ iff $x = \alpha$. If we define the class of point circuits suitably, we can find α directly from the description of $C(\alpha, \cdot)$; for instance, α is the value assigned to the input wires of C.
- Conjunctions with wild cards $C(S, \alpha, \cdot)$: the circuit $C(S, \alpha, \cdot)$ is a conjunction with wild card if it takes as input x and outputs $C(S, \alpha, x) = 1$ iff $y = \alpha$, where y is such that $y_i = x_i$ for all $i \in S$ and $y_i = 0$ for all $i \notin S$. Again, if we define this class of circuits suitably, we can find S and α directly from the description of $C(S, \alpha, \cdot)$.

On Searchability: We note that the searchability requirement in our result statement is natural and is implicit in the description of the existing constructions of copy-protection by Aaronson [3]. Another point to note is that this notion is associated with circuit classes rather than a function family.

We prove the following result. Our construction is in the common reference string (CRS) model. In this model, we assume that both the lessor and the lessee will have access to the CRS produced by a trusted setup. We note that our impossibility result also holds in the CRS model.

Theorem 2. *Assuming the existence of: (a) quantum-secure subspace obfuscators [41] and, (b) learning with errors secure against sub-exponential quantum algorithms, there exists an infinite-term secure SSL scheme in the common reference string model for searchable compute-and-compare circuits.*

Notice that for applications in which the lessor is the creator of software, the lessor can dictate how the circuit class is defined and thus would choose an implementation of the circuit class that is searchable.

On the Assumptions in Theorem 2. A discussion about the primitives described in the above theorem statement is in order. A subspace obfuscator takes as input a subspace A and outputs a circuit that tests membership of A while hiding A even against quantum adversaries. This was recently constructed by [41] based on the quantum-security of indistinguishability obfuscation [28]. Moreover, recently, there has been exciting progress in constructing quantum-secure indistinguishability obfuscation schemes [18,30,38] from cryptographic assumptions that hold against quatum adversaries.

With regards to the assumption of learning with errors against sub-exponential quantum algorithms, we firstly note that classical sub-exponential security of learning with errors has been used in the construction of many cryptographic primitives and secondly, there are no known significant quantum speedups known to solving this problem.

In the technical sections, we prove a more general theorem.

Theorem 3 (SSL for General Evasive Circuits; Informal). *Let $\mathcal{C}$ be a searchable class of circuits. Assuming the existence of: (a) quantum-secure input-hiding obfuscators [11] for $\mathcal{C}$, (b) quantum-secure subspace obfuscators [41] and, (c) learning with errors secure against sub-exponential quantum algorithms, there exists an infinite-term secure SSL scheme in the setup model for $\mathcal{C}$.*

An input-hiding obfuscator is a compiler that converts a circuit C into another functionally equivalent circuit $\widetilde{C}$ such that given $\widetilde{C}$ it is computationally hard to find an accepting point. To achieve Theorem 2, we instantiate searchable input-hiding obfuscators for compute-and-compare circuits from quantum hardness of learning with errors. However, we can envision quantum-secure instantiations of input-hiding obfuscators for more general class of searchable evasive circuits; we leave this problem open.

We admittedly use heavy cryptographic hammers to prove our result, but as will be clear in the overview given in the next section, each of these hammers will be necessary to solve the different technical challenges we face.

Concurrent Work on qVBB. Our impossibility result also rules out the existence of quantum VBB for classical circuits assuming quantum FHE and quantum learning of errors; this was stated as an open problem by Alagic and Fefferman [7]. Concurrently, [6] also rule out quantum virtual black-box obfuscation under the assumption of quantum hardness of learning with errors; unlike our work they don't additionally assume the existence of quantum FHE.

In hindsight, it shouldn't be surprising that non-black box techniques developed in the context of quantum zero-knowledge [9,16] are relevant to proving the impossibility of quantum obfuscation; the breakthrough work of Bitansky and Paneth [15] show how to construct (classical) zero-knowledge protocols with non-black box simulation using techniques developed in the context of (classical) obfuscation.

1.2 Overview of Construction of SSL

For this overview, we only focus on constructing a SSL sscheme satisfying finite-term lessor security. Our ideas can be easily adapted to the infinite-term lessor security.

To construct a SSL scheme in the setup model ($\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check}$) against arbitrary quantum poly-time (QPT) pirates, we first focus on two weaker class of adversaries, namely, *duplicators* and *maulers*. Duplicators are adversaries who, given ρ_C generated by the lessor for a circuit C sampled from a distribution $\mathcal{D}_{\mathcal{C}}$, produce $\rho_C^{\otimes 2}$; that is, all they do is replicate the state. Maulers, who given ρ_C, output $\rho_C \otimes \rho_C^*$, where ρ_C^* is far from ρ_C in trace distance and ρ_C is the copy returned by the mauler back to the lessor; that is the second copy it produces is a modified version of the original copy.

While our construction is secure against arbitrary pirates, it will be helpful to first focus on these restricted type of adversaries. We propose two schemes: the first scheme is secure against QPT maulers and the second scheme against QPT duplicators. Once we discuss these schemes, we will then show how to combine the techniques from these two schemes to obtain a construction secure against arbitrary pirates.

SSL Against Maulers. To protect SSL against a mauler, we attempt to construct a scheme using only classical cryptographic techniques. The reason why it could be possible to construct such a scheme is because maulers never produce a pirated copy ρ_C^* that is the same as the original copy ρ_C.

A natural attempt to construct a SSL scheme is to use virtual black-box obfuscation [12] (VBB): this is a compiler that transforms a circuit C into another functionally equivalent circuit $\widetilde{C}$ such that $\widetilde{C}$ only leaks the input-output behavior of C and nothing more. This is a powerful notion and implies almost all known cryptographic primitives. We generate the leased state ρ_C to be the VBB obfuscation of C, namely $\widetilde{C}$. The hope is that a mauler will not output another leased state ρ_C^* that is different from $\widetilde{C}$.

Unfortunately, this scheme is insecure. A mauler on input $\widetilde{C}$, obfuscates $\widetilde{C}$ once more to obtain $\widetilde{\widetilde{C}}$ and outputs this re-obfsuscated circuit. Moreover, note that the resulting re-obfuscated circuit still computes C. This suggests that program obfuscation is insufficient for our purpose. In hindsight, this should be unsurprising: VBB guarantees that given an obfuscated circuit, an efficient adversary should not learn anything about the implementation of the circuit, but this doesn't prevent the adversary from being able to re-produce modified copies of the obfuscated circuit.

To rectify this issue, we devise the following strategy:

- Instead of VBB, we start with a different obfuscation scheme that has the following property: given an obfuscated circuit $\widetilde{C}$, where C corresponds to an evasive function, it is computationally infeasible to determine an accepting input for C.
- We then combine this with a special proof system that guarantees the property: suppose an adversary, upon receiving $\widetilde{C}$ and a proof, outputs a *different* but functionally equivalent obfuscated circuit $\widetilde{C}^*$ along with a new proof. Then we can extract an accepting input for $\widetilde{C}$ from the adversary's proof. But this would contradict the above bullet and hence, it follows that its computationally infeasible for the adversary to output a different circuit $\widetilde{C}^*$.

To realize the above strategy, we need two separate cryptographic tools, that we define below.

Input-Hiding Obfuscators [11]: We recall the notion of input-hiding obfuscators [11]. An input-hiding obfuscator guarantees that given an obfuscated circuit $\widetilde{C}$, any efficient adversary cannot find an accepting input x, i.e., an input x such that $\widetilde{C}(x) = 1$. Of course this notion is only meaningful for an evasive class of functions: a function is evasive if given oracle access to this function, any efficient adversary cannot output an accepting point. The work of Barak et al. [11] proposed candidates for input-hiding obfuscators.

Simulation-Extractable NIZKs [26,37]: Another primitive we consider is simulation-extractable non-interactive zero-knowledge [26,37] (seNIZKs). A seNIZK system is a non-interactive protocol between a prover and a verifier with the prover trying to convince the verifier that a statement belongs to the NP language. By non-interactive we mean that the prover only sends one message to the verifier and the verifier is supposed to output the decision bit: accept or reject. Moreover, this primitive is defined in the common reference string model. In this model, there is a trusted setup that produces a common reference string and both the prover and the verifier have access to this common reference string.

As in a traditional interactive protocol, we require a seNIZK to satisfy the completeness property. Another property we require is simulation-extractability. Simulation-extractability, a property that implies both zero-knowledge and soundness, guarantees that if there exists an efficient adversary $\mathcal{A}$ who upon receiving a *simulated* proof[10] for an instance x, produces an accepting proof for a different instance x', i.e., $x' \neq x$, then there also exists an adversary $\mathcal{B}$ that given the same simulated proof produces an accepting proof for x' along with simultaneously producing a valid witness for x'.

[10] A simulated proof is one that is generated by an efficient algorithm, called a simulator, who has access to some private coins that was used to generate the common reference string. Moreover, a simulated proof is indistinguishable from an honestly generated proof. A simulator has the capability to generate simulated proofs for YES instances even without knowing the corresponding witness for these instances.

Combining Simulation-Extractable NIZKs and Input-Hiding Obfuscators: We now combine the two tools we introduced above to obtain a SSL scheme secure against maulers. Our SSL scheme will be associated with searchable circuits; given a description of a searchable circuit C, an input x can be determined efficiently such that $C(x) = 1$.

To lease a circuit C, do the following:

- Compute an input-hiding obfuscation of C, denoted by $\widetilde{C}$,
- Produce a seNIZK proof π that proves knowledge of an input x such that $C(x) = 1$. Note that we can find this input using the searchability property.

Output $(\widetilde{C}, \pi)$ as the leased circuit. To evaluate on any input x, we first check if π is a valid proof and if so, we compute $\widetilde{C}$ on x to obtain $C(x)$.

To see why this scheme is secure against maulers, suppose an adversary $\mathcal{A}$ given $(\widetilde{C}, \pi)$ produces $(\widetilde{C}^*, \pi^*)$, where $\widetilde{C}^* \neq \widetilde{C}$. Since $\mathcal{A}$ is a valid mauler we are guaranteed that $\widetilde{C}^*$ is functionally equivalent to C. We first run the seNIZK simulator to simulate π and once this is done, we no longer need x to generate π. Now, we invoke the simulation-extractability property to convert $\mathcal{A}$ into one who not only produces $(\widetilde{C}^*, \pi^*)$ but also simultaneously produces x such that $\widetilde{C}^*(x) = 1$. Since $\widetilde{C}^*$ is functionally equivalent to C, it follows that $C(x) = 1$ as well. But this violates the input-hiding property which says that no efficient adversary given $\widetilde{C}$ can produce an accepting input.

Issue: Checking Functional Equivalence. There is a subtlety we skipped in the proof above. The maulers that we consider have multi-bit output which is atypical in the cryptographic setting where the focus is mainly on boolean adversaries. This causes an issue when we switch from the honestly generated proof to a simulated proof. Upon receiving the honestly generated proof, $\mathcal{A}$ outputs $(\widetilde{C}^*, \pi^*)$ such that $\widetilde{C}^*$ is functionally equivalent to C but upon receiving the simulated proof, the adversary outputs $(\widetilde{C}^*, \pi^*)$ where $\widetilde{C}^*$ differs from C on one point. From $\mathcal{A}$, we need to extract one bit that would help distinguish the real and simulated proofs. To extract this bit, we rely upon sub-exponential security. Given $\widetilde{C}^*$, we run in time 2^n, where n is the input length, and check if $\widetilde{C}^*$ is still functionally equivalent to C; if indeed $\widetilde{C}^*$ is not functionally equivalent to C then we know for a fact that the adversary was given a simulated proof, otherwise it received an honestly generated proof. We set the security parameter in the seNIZK system to be sufficiently large (for eg, poly(n)) such that the seNIZK is still secure against adversaries running in time 2^n.

SSL Against Duplicators. Next we focus on constructing SSL secure against duplicators. If our only goal was to protect against duplicators, we could achieve this with a simple scheme. The lessor, in order to lease C, will output $(|\psi\rangle, C)$ where $|\psi\rangle$ is a random quantum state generated by applying a random polynomial sized quantum circuit U on input $|0^{\otimes \lambda}\rangle$. Run on input $(|\psi\rangle, C, x)$ ignores the quantum state $|\psi\rangle$, and outputs $C(x)$. By quantum no-cloning, an attacker cannot output two copies of $(|\psi\rangle, C)$, which means that this scheme is already secure against duplicators.

Recall that we focused on designing SSL for duplicators in the hope that it will be later helpful for designing SSL for arbitrary pirates. But any SSL scheme in which Run ignores the quantum part would not be useful for obtaining SSL secure against arbitrary pirates; an attacker can simply replace the quantum state as part of the leased state with its own quantum state and copy the classical part. To overcome this insufficiency, we need to design SSL schemes where the Run algorithm only computes correctly when the input leased state belongs to a sparse set of quantum states. This suggests that the Run algorithm implicitly satisfies a verifiability property; it should be able to verify that the input quantum state lies in this sparse set.

Publicly Verifiable Unclonable States. We wish to construct a family of efficiently preparable states $\{|\psi_s\rangle\}_s$ with the following verifiability property. For any state $|\psi_s\rangle$ in the family, there is a way to sample a classical description d_s for $|\psi_s\rangle$ in such a way that it can be verified that d_s is a corresponding description of $|\psi_s\rangle$. To be more precise, there should be a verification algorithm $\mathsf{Ver}(|\psi_s\rangle, d)$ that accepts if d is a valid description for $|\psi_s\rangle$. Furthermore, we want the guarantee that given a valid pair $(|\psi_s\rangle, d_s)$, no QPT adversary can produce $|\psi_s\rangle^{\otimes 2}$.

Our requirement has the same flavor as public-key quantum money, but a key difference is that we do not require any secret parameters associated with the scheme. Moreover, we allow anyone to be able to generate such tuples $(|\psi_s\rangle, d_s)$ and not just the minting authority (bank).

Given such verifiable family, we can define the Run algorithm as follows,

$\mathsf{Run}(C, (|\psi_s\rangle, d), x)$:

- If $\mathsf{Ver}(|\psi_s\rangle, d) = 0$, output $\bot$.
- Otherwise, output $C(x)$.

Any lessor can now lease a state $(|\psi_s\rangle, d_s, C)$, which would allow anyone to compute C using Run. Of course, any pirate that is given $(|\psi_s\rangle, d_s, C)$ can prepare their own $(|\psi_{s'}\rangle, d_{s'})$ and then input $(|\psi_{s'}\rangle, d_{s'}, C)$ into Run. But recall that we are only interested in ruling out *duplicators*. From the public verifiable property of the quantum states, we have the fact that no QPT pirate could prepare $|\psi_s\rangle^{\otimes 2}$ from $(|\psi_s\rangle, d_s)$ and thus, it is computationally infeasible to duplicate the leased state.

Publicly Verifiable Unclonable States from Subspace Hiding Obfuscation. The notion of publicly verifiable unclonable states was first realized by Zhandry [41]. The main tool used in Zhandry's construction is yet another notion of obfuscation, called subspace hiding obfuscation. Roughly speaking, a subspace hiding obfuscator (shO) takes as input a description of a linear subspace A, and outputs a circuit that computes the membership function for A, i.e. $\mathsf{shO}(A)(x) = 1$ iff $x \in A$. Zhandry shows that for a uniformly random $\frac{\lambda}{2}$-dimensional subspace $A \subset \mathbb{Z}_q^\lambda$, given $|A\rangle := \frac{1}{\sqrt{q^{\lambda/2}}} \sum_{a \in A} |a\rangle$ along with $\widetilde{g} \leftarrow \mathsf{shO}(A), \widetilde{g_\perp} \leftarrow \mathsf{shO}(A^\perp)$, no QPT algorithm can prepare $|A\rangle^{\otimes 2}$ with non-negligible probability. Neverthe-

less, because $\widetilde{g}$ and $\widetilde{g_\perp}$ compute membership for A and $A^\perp$ respectively, it is possible to project onto $|A\rangle\langle A|$ using $(\widetilde{g}, \widetilde{g_\perp})$. This lets anyone check the tuple $(|\psi\rangle, (\widetilde{g}, \widetilde{g_\perp}))$ by measuring $|\psi\rangle$ with the projectors $\{|A\rangle\langle A|, I - |A\rangle\langle A|\}$.

Main Template: SSL Against Pirates. Our goal is to construct SSL against arbitrary QPT pirates and not just duplicators or maulers. To achieve this goal, we combine the techniques we have developed so far.

To lease a circuit C, do the following:

1. First prepare the state the state $|A\rangle = \frac{1}{\sqrt{q^{\lambda/2}}} \sum_{a \in A} |a\rangle$, along with $\tilde{g} \leftarrow \mathsf{shO}(A)$ and $\widetilde{g_\perp} \leftarrow \mathsf{shO}(A^\perp)$.
2. Compute an input-hiding obfuscation of C, namely $\widetilde{C}$.
3. Let x be an accepting point of C. This can be determined using the searchability condition.
4. Compute a seNIZK proof π such that: (1) the obfuscations $(\widetilde{g}, \widetilde{g_\perp}, \widetilde{C})$ were computed correctly, as a function of $(A, A^\perp, C)$, and, (2) $C(x) = 1$.
5. Output $|\psi_C\rangle = (|A\rangle, \widetilde{g}, \widetilde{g_\perp}, \widetilde{C}, \pi)$.

The Run algorithm on input $(|\psi_C\rangle, \widetilde{g}, \widetilde{g_\perp}, \widetilde{C}, \pi)$ and x, first checks the proof π, and outputs $\perp$ if it does not accept the proof. If it accepts the proof, it knows that $\widetilde{g}$ and $\widetilde{g_\perp}$ are subspace obfuscators for some subspaces A and $A^\perp$ respectively; it can use them to project $|\psi_C\rangle$ onto $|A\rangle\langle A|$. This checks whether $|\psi_C\rangle$ is the same as $|A\rangle$ or not. If it is not, then it outputs $\perp$. If it has not output $\perp$ so far, then it computes $\widetilde{C}$ on x to obtain $C(x)$.

Proof Intuition: To prove the lessor security of the above scheme, we consider two cases depending on the behavior of the pirate:

- *Duplicator*: in this case, the pirate produces a new copy that is of the form $(\sigma^*, \widetilde{g}, \widetilde{g_\perp}, \widetilde{C}, \pi)$; that is, it has the same classical part as before. If σ^* is close to $|A\rangle\langle A|$, it would violate the no-cloning theorem. On the other hand, if σ^* is far from $|A\rangle\langle A|$, we can argue that the execution of Run on the copy produced by the pirate will not compute C. The reason being that at least one of the two subspace obfuscators $\widetilde{g}, \widetilde{g_\perp}$ will output $\perp$ on the state σ^*.
- *Mauler*: suppose the pirate produces a new copy that is of the form $(\sigma^*, \widetilde{g}^*, \widetilde{g_\perp}^*, \widetilde{C}^*, \pi^*)$ such that $(\widetilde{g}^*, \widetilde{g_\perp}^*, \widetilde{C}^*) \neq (\widetilde{g}, \widetilde{g_\perp}, \widetilde{C})$. We invoke the simulation-extractability property to find an input x such that $\widetilde{C}^*(x) = 1$. Since $\widetilde{C}^*$ is assumed to have the same functionality as C, this means that $C(x) = 1$. This would contradict the security of input-hiding obfuscation, since any QPT adversary even given $\widetilde{C}$ should not be able to find an accepting input x such that $C(x) = 1$.

Organization. We provide the related works and preliminary background in the full version. We present the formal definition of secure software leasing in Sect. 2. The impossibility result is presented in Sect. 3. Finally, we present the positive result in Sect. 4.

2 Secure Software Leasing (SSL)

We present the definition of secure software leasing schemes. A secure software leasing (SSL) scheme for a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following QPT algorithms.

- **Private-key Generation,** $\mathsf{Gen}(1^\lambda)$: On input security parameter λ, outputs a private key sk.
- **Software Lessor,** $\mathsf{Lessor}\,(\mathrm{sk}, C)$: On input the private key sk and a poly(n)-sized classical circuit $C \in \mathcal{C}_\lambda$, with input length n and output length m, outputs a quantum state ρ_C.
- **Evaluation,** $\mathsf{Run}(\rho_C, x)$: On input the quantum state ρ_C and an input $x \in \{0,1\}^n$, outputs y, and some state $\rho'_{C,x}$.
- **Check of Returned Software,** $\mathsf{Check}\,(\mathrm{sk}, \rho^*_C)$: On input the private key sk and the state ρ^*_C, it checks if ρ^*_C is a valid leased state and if so it outputs 1, else it outputs 0.

Setup. In this work, we only consider SSL schemes in the setup model. In this model, all the lessors in the world have access to a common reference string generated using a PPT algorithm Setup. The difference between Setup and Gen is that Setup is run by a trusted third party whose output is used by all the lessors while Gen is executed by each lessor separately. We note that our impossibility result rules out SSL schemes for all quantum unlearnable class of circuits even in the setup model.

We define this notion below.

Definition 1 (SSL with Setup). *A secure software leasing scheme* $(\mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ *is said to be in the common reference string (CRS) model if additionally, it has an algorithm* Setup *that on input* 1^λ *outputs a string* crs.

Moreover, the algorithm Gen *now takes as input* crs *instead of* 1^λ *and* Run *additionally takes as input* crs.

We require that a SSL scheme, in the setup model, satisfies the following properties.

Definition 2 (Correctness). *A SSL scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ *for* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* ε*-correct if for all* $C \in \mathcal{C}_\lambda$*, with input length* n*, the following two properties holds for some negligible function* ε*:*

- *Correctness of Run:*

$$\Pr\left[\forall x \in \{0,1\}^n,\ y = C(x) \ :\ \begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\ \mathrm{sk}\leftarrow\mathsf{Gen}(\mathsf{crs}),\\ \rho_C\leftarrow\mathsf{Lessor}(\mathrm{sk},C)\\ (\rho'_{C,x},y)\leftarrow\mathsf{Run}(\mathsf{crs},\rho_C,x)\end{array}\right] \geq 1-\varepsilon$$

- *Correctness of Check:*

$$\Pr\left[\mathsf{Check}\,(\mathrm{sk},\rho_C)=1 \ :\ \begin{array}{c}\mathsf{crs}\leftarrow\mathsf{Setup}(1^\lambda),\\ \mathrm{sk}\leftarrow\mathsf{Gen}(\mathsf{crs})\\ \rho_C\leftarrow\mathsf{Lessor}(\mathrm{sk},C)\end{array}\right] \geq 1-\varepsilon$$

Reusability. A desirable property of a SSL scheme is reusability: the lessee should be able to repeatedly execute Run on multiple inputs. A SSL scheme does not necessarily guarantee reusability; for instance, Run could destroy the state after executing it just once. But fortunately, we can transform this scheme into another scheme that satisfies reusability.

We define reusability formally.

Definition 3. *(Reusability) A SSL scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ *for* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *is said to be reusable if for all* $C \in \mathcal{C}$ *and for all* $x \in \{0,1\}^n$,

$$\left\|\rho'_{C,x} - \rho_C\right\|_{tr} \leq negl(\lambda).$$

Note that the above requirement $\left\|\rho'_{C,x} - \rho_C\right\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda)$ would guarantee that an evaluator can evaluate the leased state on multiple inputs; on each input, the original leased state is only disturbed a little which means that the resulting state can be reused for evaluation on other inputs.

The following proposition states that any SSL scheme can be converted into one that is reusable.

Proposition 2. *Let* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ *be any SSL scheme (not necessarily satisfying the reusability condition). Then, there is a QPT algorithm* Run' *such that* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}', \mathsf{Check})$ *is a reusable SSL scheme.*

Proof. For any $C \in \mathcal{C}$ and for any $x \in \{0,1\}^n$, we have that $\mathsf{Run}(\mathsf{crs}, \rho_C, x)$ outputs $C(x)$ with probability $1 - \varepsilon$. By the Almost As Good As New Lemma (see full version), there is a way to implement Run such that it is possible to obtain $C(x)$, and then recover a state $\widetilde{\rho_C}$ satisfying $\|\widetilde{\rho_C} - \rho_C\|_{\mathrm{tr}} \leq \sqrt{\varepsilon}$. We let Run' be this operation.

Thus, it suffices to just focus on the correctness property when constructing a SSL scheme.

2.1 Security

Our notion intends to capture the different scenarios discussed in the introduction. In particular, we want to capture the security guarantee that given an authorized (valid) copy ρ_C, no pirate can output two authorized copies. We will assume that these valid copies contain a quantum state and a classical string. The Run algorithm expects valid copies to have this form; without loss of generality, the classical part can always be measured before executing Run.

Finite-Term Lessor Security. We require the following security guarantee: suppose a QPT adversary (pirate) receives a leased copy of C generated using Lessor; denote this by ρ_C. We require that the pirate cannot produce a bipartite state σ^* on registers R_1 and R_2, such that $\sigma_1^* := \mathrm{Tr}_2[\sigma^*]$ passes the verification by Check, and the resulting *post-measurement* state on R_2, which we denote by $P_2(\sigma^*)$, still computes C by $\mathsf{Run}(P_2(\sigma^*), x) = C(x)$.

Before formally stating the definition, let us fix some notation. We will use the following notation for the state that the pirate keeps after the initial copy has been returned and verified. If the pirate outputs the bipartite state σ^*, then we will write

$$P_2(\mathsf{sk}, \sigma^*) \propto \mathrm{Tr}_1 \left[\Pi_1[\mathsf{Check}(\mathsf{sk}, \cdot)_1 \otimes I_2 (\sigma^*)] \right]$$

for the state that the pirate keeps *after* the first register has been returned and verified. Here, Π_1 denotes projecting the output of Check onto 1, and where $\mathsf{Check}(\mathsf{sk}, \cdot)_1 \otimes I_2(\sigma^*)$ denotes applying the Check QPT onto the first register, and the identity on the second register of σ^*. In other words, $P_2(\mathsf{sk}, \sigma^*)$ is used to denote the post-measurement state on R_2 conditioned on $\mathsf{Check}(\mathsf{sk}, \cdot)$ accepting on R_1.

Definition 4 (Finite-Term Perfect Lessor Security). *We say that a SSL scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ *for a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *is said to satisfy* $(\beta, \gamma, \mathcal{D}_\mathcal{C})$**-perfect finite-term lessor security**, *with respect to a distribution* $\mathcal{D}_\mathcal{C}$ *on* $\mathcal{C}$*, if for every QPT adversary* $\mathcal{A}$ *(pirate) that outputs a bipartite (possibly entangled) quantum state on two registers,* R_1 *and* R_2*, the following holds:*

$$\Pr\left[\begin{array}{c} \mathsf{Check}(\mathrm{sk},\sigma_1^*)=1 \\ \wedge \\ \forall x,\ \Pr[\mathsf{Run}(\mathsf{crs},P_2(\mathsf{sk},\sigma^*),x)=C(x)]\geq\beta \end{array} : \begin{array}{c} \mathsf{crs}\leftarrow\mathsf{Setup}\left(1^\lambda\right), \\ C\leftarrow\mathcal{D}_\mathcal{C}(\lambda), \\ \mathrm{sk}\leftarrow\mathsf{Gen}(\mathsf{crs}), \\ \rho_C\leftarrow\mathsf{Lessor}(\mathrm{sk},C), \\ \sigma^*\leftarrow\mathcal{A}(\mathsf{crs},\rho_C) \\ \sigma_1^*=Tr_2[\sigma^*] \end{array}\right] \leq \gamma$$

Remark 1. The reason why we use the word perfect here is because we require $\mathsf{Run}(P_2(\sigma^*), x) = C(x)$ to hold with probability at least β on *every* input x. Note that Run is not necessarily deterministic (for instance, it could perform measurements) and thus we allow it to output the incorrect value with some probability.

2.2 Infinite-Term Lessor Security

In the infinite-term lease case, we want the following security notion: given (σ_1^*, σ_2^*) generated by a pirate $\mathcal{A}(\rho_C)$, guarantees that if one copy satisfies the correctness,

$$\forall x \Pr[\mathsf{Run}(\mathsf{crs}, \sigma_1^*, x) = C(x)] \geq \beta$$

for some non-negligible β, then after successfully evaluating $C(x)$ using σ_1^* on any input x^*, it should be the case that the resulting state on the second register, which we will denote by $\mathcal{E}_{x^*}^{(2)}(\sigma^*)$, cannot also satisfy

$$\forall x \Pr[\mathsf{Run}(\mathsf{crs}, \mathcal{E}_{x^*}^{(2)}(\sigma^*), x) = C(x)] \geq \beta.$$

In other words, if one of the copies has already been succesful in computing C in Run, then there will be inputs in which the second copy cannot evaluate C with better than negligible probability.

This security notion would rule out the following scenario. Eve gets a copy of ρ_C and gives σ_1^* to Alice and σ_2^* to Bob. Alice now chooses an input x_A, and Bob an input x_B. It cannot be the case that for all inputs (x_A, x_B) they choose, they will compute $(C(x_A), C(x_B))$ with non-negligible probability.

Definition 5 (Infinite-term Perfect Lessor Security). *We say that a SSL scheme* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ *for a class of circuits* $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$ *is said to be* $(\gamma, \beta, \mathcal{D}_{\mathcal{C}})$-***infinite-term perfect lessor secure****, with respect to a distribution* $\mathcal{D}_{\mathcal{C}}$*, if for every QPT adversary* $\mathcal{A}$ *(pirate) that outputs a bipartite (possibly entangled) quantum state on two registers,* R_1 *and* R_2*, the following holds:*

$$\Pr\left[\forall x, \begin{pmatrix} \Pr[(\mathsf{Run}(\mathsf{crs}, x, \sigma_1^*) = C(x)] \geq \beta \\ \wedge \\ \forall x', \Pr\left[\mathsf{Run}(\mathsf{crs}, x', \mathcal{E}_x^{(2)}(\sigma^*)) = C(x')\right] \geq \beta \end{pmatrix} : \begin{matrix} \mathsf{crs} \leftarrow \mathsf{Setup}\left(1^\lambda\right), \\ C \leftarrow \mathcal{D}_{\mathcal{C}}(\lambda), \\ \mathrm{sk} \leftarrow \mathsf{Gen}(\mathsf{crs}), \\ \rho_C \leftarrow \mathsf{Lessor}(\mathrm{sk}, C), \\ \sigma^* \leftarrow \mathcal{A}(\mathsf{crs}, \rho_C) \\ \sigma_1^* = Tr_2[\sigma^*] \end{matrix}\right] \leq \gamma.$$

Remark 2. Both finite and infinite-term security can be extended to the case where the pirate is given multiple copies, $\rho_C^{\otimes m}$, where ρ_C is the output of Lessor on C. In the finite-term case, we require the following: if a pirate outputs $m+1$ copies and moreover, the m initial copies are returned and succesfully checked, computing Run on the remaining copy (that the pirate did not return) will not be functionally equivalent to the circuit C. In the infinite-term case, the pirate cannot output $m+1$ copies where Run on each of the $m+1$ copies can be used to successfully compute C.

3 Impossibility of SSL

To prove the impossibility of SSL, we first construct *de-quantumizable* class of circuits.

3.1 De-Quantumizable Circuits: Definition

A de-quantumizable class of circuits $\mathcal{C}$ is a class of circuits for which there is a QPT algorithm that given any quantum circuit with the same functionality as $C \in \mathcal{C}$, it finds a (possibly different) classical circuit $C' \in \mathcal{C}$ with the same functionality as C. Of course if $\mathcal{C}$ is learnable, then it could be possible to just observe the input-output behavior of the quantum circuit to find such a C'. To make this notion meaningful, we additionally impose the requirement that $\mathcal{C}$ needs to be quantum unlearnable; given only oracle access to C, any quantum algorithm can find a circuit (possibly a quantum circuit and an auxiliary input state ρ) with the same functionality as C with only negligible probability.

Definition 6. *We say that a collection of QPT algorithms, $\{U_C, \rho_C\}_{C\in\mathcal{C}}$, computes $\mathcal{C}$ if for any $C \in \mathcal{C}$, with input length n and output length m, ρ_C is a* poly(n)*-qubits auxiliary state, and U_C a QPT algorithm satisfying that for all $x \in \{0,1\}^n$,*

$$\mathsf{Pr}[U_C(\rho_C, x) = C(x)] \geq 1 - negl(\lambda),$$

where the probability is over the measurement outcomes of U_C. We also refer to (U_C, ρ_C) as an efficient quantum implementation of C. A class of classical circuits $\mathcal{C}$, associated with a distribution $\mathcal{D}_{\mathcal{C}}$, is said to be de-quantumizable *if the following holds:*

- **Efficient de-quantumization***: There is a QPT algorithm $\mathcal{B}$ such that, for any $\{U_C, \rho_C\}_{C\in\mathcal{C}}$ that computes $\mathcal{C}$, the following holds:*

$$\mathsf{Pr}\left[\begin{array}{c} C'\in\mathcal{C} \\ \wedge \\ \forall x\in\{0,1\}^n, C(x)=C'(x) \end{array} \;:\; \begin{array}{c} C\leftarrow\mathcal{D}_{\mathcal{C}} \\ C'(x)\leftarrow\mathcal{B}(U_C,\rho_C) \end{array}\right] \geq 1 - negl(\lambda)$$

- **ν-Quantum Unlearnability:** *For any QPT adversary $\mathcal{A}$, the following holds:*

$$\mathsf{Pr}\left[\forall x, \mathsf{Pr}[U^*(\rho^*, x) = C(x)] \geq \nu \;:\; \begin{array}{c} C\leftarrow\mathcal{D}_{\mathcal{C}} \\ (U^*,\rho^*)\leftarrow\mathcal{A}^{C(\cdot)}(1^\lambda) \end{array}\right] \leq negl(\lambda)$$

Remark 3. By the Almost As Good As New Lemma (we present the lemma in the full version), we can assume that the QPT algorithm U_C also output a state $\rho'_{C,x}$ that is negligibly close in trace distance to ρ_C, i.e. for all $C \in \mathcal{C}$ and $x \in \{0,1\}^n$ it holds that

$$\mathsf{Pr}[U_C(\rho_C, x) = (\rho'_{C,x}, C(x))] \geq 1 - \mathrm{negl}(\lambda)$$

and $\left\|\rho'_{C,x} - \rho_C\right\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda)$.

Remark 4. We emphasize that the efficient de-quantumization property requires that the circuit C' output by the adversary should be in the same circuit class $\mathcal{C}$.

Remark 5. We can relax the unlearnability condition in the above definition to instead have a distribution over the inputs and have the guarantee that the adversary has to output a circuit (U^*, ρ^*) such that it agrees with C only on inputs drawn from this distribution. Our impossibility result will also rule out this relaxed unlearnability condition; however, for simplicity of exposition, we consider the unlearnability condition stated in the above definition.

From the above definition, we can see why a de-quantumizable class $\mathcal{C}$ cannot be copy-protected, as there is a QPT $\mathcal{B}$ that takes any (U_C, ρ_C) efficiently computing C, and outputs a functionally equivalent *classical* circuit C', which can be copied. In the following theorem we will show that if every circuit $C \in \mathcal{C}$ have a unique representation in $\mathcal{C}$, then it is also not possible to have SSL for this circuit class. To see why we need an additional condition, lets consider a QPT pirate $\mathcal{A}$ that wants to break SSL given (Run, ρ_C) computing $C \in \mathcal{C}$. Then, $\mathcal{A}$ can

run $\mathcal{B}$ to obtain a circuit $C' \in \mathcal{C}$, but in the proccess it could have destroyed ρ_C, hence it wouldn't be able to return the initial copy. If $\mathcal{B}$ takes as input (Run, ρ_C) and outputs a *fixed* C' with probability neglibly close to 1, then by the Almost As Good As New Lemma, it could uncompute and recover ρ_C. The definition of de-quantumizable class does not guarantee that $\mathcal{B}$ will output a fixed circuit C', unless each circuit in the family has a unique representation in $\mathcal{C}$. If each circuit has a unique representation, the pirate would obtain $C' = C$ with probability neglibly close to 1, and uncompute to recover ρ_C. At this point, the pirate can generate its own leasing keys sk', and run $\mathsf{Lessor}(\mathsf{sk}', C')$ to obtain a valid leased state $\rho'_{C'}$. The pirate was able to generate a new valid leased state for C, while preserving the initial copy ρ_C, which it can later return to the lessor.

Theorem 4. *Let $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$ be a de-quantumizable class of circuits in which every circuit in the support of $\mathcal{D}_{\mathcal{C}}$ has a unique representation in $\mathcal{C}$. Then there is no SSL scheme $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Lessor}, \mathsf{Run}, \mathsf{Check})$ (in CRS model) for $\mathcal{C}$ satisfying ε-correctness and $(\beta, \gamma, \mathcal{D}_{\mathcal{C}})$-perfect finite-term lessor security for any negligible γ, and any $\beta \leq (1 - \varepsilon)$.*

Proof. Consider the QPT algorithm $\mathcal{A}$ (pirate) that is given $\rho_C \leftarrow \mathsf{Lessor}(\mathrm{sk}, C)$ for some $C \leftarrow \mathcal{D}_{\mathcal{C}}$. The pirate will run $\mathcal{B}$, the QPT that de-quantumizes $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$, on input (Run, ρ_C) to obtain a functionally equivalent circuit $C' \in \mathcal{C}$. Because C has a unique representation in $\mathcal{C}$, we have $C' = C$. Since this succeeds with probability neglibly close to 1, by the Almost As Good As New Lemma, it can all be done in a way such that it is possible to obtain C and to recover a state $\widetilde{\rho_C}$ satisfying $\|\widetilde{\rho_C} - \rho_C\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda)$. At this point, the pirate generates its own key $\mathrm{sk}' \leftarrow \mathsf{Gen}(\mathsf{crs})$, and prepares $\rho'_C \leftarrow \mathsf{Lessor}(\mathrm{sk}', C)$. It outputs $\widetilde{\rho_C} \otimes \rho'_C$.

This means that ρ'_C is a valid leased state and by correctness of the SSL scheme,

$$\Pr\left[\forall x \in \{0,1\}^n,\ \mathsf{Run}\left(\mathsf{crs}, \rho'_C, x\right) = C(x) \ : \ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}\left(1^\lambda\right), \\ \mathrm{sk}' \leftarrow \mathsf{Gen}(\mathsf{crs}), \\ \rho'_C \leftarrow \mathsf{Lessor}(\mathrm{sk}', C) \end{array}\right] \geq 1 - \varepsilon$$

Furthermore, since $\|\widetilde{\rho_C} - \rho_C\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda)$, the probability that $\widetilde{\rho_C}$ passes the return check is neglibly close to 1. Putting these together, we have

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{Check}(\mathrm{sk}, \widetilde{\rho_C}) = 1 \\ \wedge \\ \forall x,\ \Pr\left[\mathsf{Run}(\mathsf{crs}, \rho'_C, x) = C(x)\right] \geq 1 - \varepsilon \end{array} \ : \ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}\left(1^\lambda\right), \\ C \leftarrow \mathcal{D}_{\mathcal{C}}(\lambda), \\ \mathrm{sk} \leftarrow \mathsf{Gen}(\mathsf{crs}), \\ \rho_C \leftarrow \mathsf{Lessor}(\mathrm{sk}, C), \\ \widetilde{\rho_C} \otimes \rho'_C \leftarrow \mathcal{A}(\mathsf{crs}, \rho_C) \end{array}\right] \geq 1 - \mathrm{negl}(\lambda)$$

3.2 De-Quantumizable Circuit Class: Construction

All that remains in the proof of impossibility of SSL is the construction of a de-quantumizable circuits class $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$ in which every circuit in the support of $\mathcal{D}_{\mathcal{C}}$ has a unique representation in $\mathcal{C}$. We begin with an overview of the construction.

Constructing de-quantumizable Circuits: Challenges. The starting point is the seminal work of Barak et al. [12], who demonstrated a class of functions, where each function is associated with a secret key k, such that: (a) *Non-black-box secret extraction*: given non-black-box access to any classical circuit implementation of this function, the key can be efficiently recovered, (b) *Classical Unlearnability of secrets*: but given black-box access to this circuit, any classical adversary who can only make polynomially many queries to the oracle cannot recover the key.

While the result of Barak et al. has the ingredients suitable for us, it falls short in many respects:

- The proof of non-black-box secret extraction crucially relies upon the fact that we are only given a classical obfuscated circuit. In fact there are inherent difficulties that we face in adapting Barak et al. to the quantum setting; see [7].
- As is the case with many black-box extraction techniques, the proof of Barak et al. involves evaluating the obfuscated circuit multiple times in order to recover the secret. As is typically the case with quantum settings, evaluating the same circuit again and again is not always easy – the reason being that evaluating a circuit once could potentially destroy the state thus rendering it impossible to run it again.
- Barak et al. only guarantees extraction of secrets given black-box access to the classical circuit implementation of the function. However, our requirement is qualitatively different: given a quantum implementation of the classical circuit, we need to find a (possible different) classical circuit with the same functionality.
- Barak et al.'s unlearnability result only ruled out adversaries who make classical queries to the oracle. On the other hand, we need to argue unlearnability against QPT adversaries who can perform superposition queries to the oracle.

Nonetheless, we show that the techniques introduced in a simplified version of Barak[11] can be suitably adapted for our purpose by using two tools: quantum fully homomorphic encryption (QFHE) and lockable obfuscation. Combining QFHE and lockable obfuscation for the purpose of secret extraction has been recently used in a completely different context, that of building zero-knowledge protocols [9,16] (and in classical setting was first studied by [14]).

Construction. We present the construction of de-quantumizable circuits.

Theorem 5. *Assuming the quantum hardness of learning with errors (QLWE), and assuming that there is a* QFHE *that supports evaluation of arbitrary polynomial-sized quantum circuits, and has the following two properties: (a) ciphertexts have classical plaintexts have classical descriptions and, (b) classical ciphertexts can be decrypted using a classical circuit,*

there exists a de-quantumizable class of circuits $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$.

[11] See [15] for a description of this simplified version.

Proof. We define a de-quantumizable class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where every circuit in $\mathcal{C}_\lambda$ is defined as follows:

$C_{a,b,r,\mathsf{pk},\mathcal{O}}(x)$:

1. If $x = 0 \cdots 0$, output $\mathsf{QFHE.Enc}(\mathsf{pk}, a; r) |\mathcal{O}|\mathsf{pk}$.
2. Else if $x = a$, output b.
3. Otherwise, output $0 \cdots 0$

We will suitably pad with zeroes such that all the inputs (resp., outputs) are of the same length n (resp., of the same length m).

Let $\mathcal{D}_{\mathcal{C}}(\lambda)$ be the distribution that outputs a circuit from $\mathcal{C}_\lambda$ by sampling $a, b, r \xleftarrow{\$} \{0,1\}^\lambda$, then computing $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{QFHE.Gen}(1^\lambda)$, and finally computing an obfuscation $\mathcal{O} \leftarrow \mathsf{LO.Obf}(\mathbf{C}[\mathsf{QFHE.Dec}(\mathsf{sk}, \cdot), b, (\mathsf{sk}|r)])$, where $\mathbf{C}$ is a compute-and-compare circuit.

We show that with respect to this distribution: (a) $\mathcal{C}$ is quantum unlearnable (Proposition 3) and, (b) $\mathcal{C}$ is efficiently de-quantumizable (Proposition 4).

Proposition 3. *For any non-negligible ν, the circuit class $\mathcal{C}$ is ν-quantum unlearnable with respect to $\mathcal{D}_{\mathcal{C}}$.*

We provide a proof of the above proposition in the full version.

Proposition 4. *$(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$ is efficiently de-quantumizable.*

Proof. We will start with an overview of the proof.

Overview: Given a quantum circuit (U_C, ρ_C) that computes $C_{a,b,r,\mathsf{pk},\mathcal{O}}(\cdot)$, first compute on the input $x = 0 \cdots 0$ to obtain $\mathsf{QFHE.Enc}(\mathsf{pk}, a; r)|\mathcal{O}|\mathsf{pk}$. We then homomorphically evaluate the quantum circuit on $\mathsf{QFHE.Enc}(\mathsf{pk}, a; r)$ to obtain $\mathsf{QFHE.Enc}(\mathsf{pk}, b')$, where b' is the output of the quantum circuit on input a; this is part where we crucially use the fact that we are given (U_C, ρ_C) and not just black-box access to the functionality computing (U_C, ρ_C). But b' is nothing but b! Given QFHE encryption of b, we can then use the lockable obfuscation to recover sk; since the lockable obfuscation on input a valid encryption of b outputs sk. Using sk we can then recover the original circuit $C_{a,b,r,\mathsf{pk},\mathcal{O}}(\cdot)$. Formal details follow.

For any $C \in \mathcal{C}$, let (U_C, ρ_C) be any QPT algorithm (with auxiliary state ρ_C) satisfying that for all $x \in \{0,1\}^n$,

$$\Pr\left[U_C(\rho_C, x) = \left(\rho'_{C,x}, C(x)\right)\right] \geq 1 - \mathrm{negl}(\lambda),$$

where the probability is over the measurement outcomes of U_C, and $\rho'_{C,x}$ is negligibly close in trace distance to ρ_C (see Remark 3). We will show how to constuct a QPT $\mathcal{B}$ to de-quantumize $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$.

$\mathcal{B}$ will perform a QFHE evaluation, which we describe here. Given $\mathsf{QFHE.Enc}(\mathsf{pk}, x)$, we want to homomorphically evaluate $C(x)$ to obtain

$\mathsf{QFHE.Enc}(\mathsf{pk}, C(x))$. To do this, first prepare $\mathsf{QFHE.Enc}(\mathsf{pk}, \rho_C, x)$, then evaluate U_C homomorphically to obtain the following:

$$\mathsf{QFHE.Enc}(\mathsf{pk}, \rho'_{C,x}, C(x)) = \mathsf{QFHE.Enc}(\mathsf{pk}, \rho'_{C,x}) \big| \mathsf{QFHE.Enc}(\mathsf{pk}, C(x))$$

Consider the following QPT algorithm $\mathcal{B}$ that is given (U_C, ρ_C) for any $C \in \mathcal{C}$. $\mathcal{B}(U_C, \rho_C)$:

1. Compute $(\rho', \mathsf{ct}_1 | \mathcal{O}' | \mathsf{pk}') \leftarrow U_C(\rho_C, 0 \cdots 0)$.
2. Compute $\sigma | \mathsf{ct}_2 \leftarrow \mathsf{QFHE.Eval}(U_C(\rho', \cdot), \mathsf{ct}_1)$
3. Compute $\mathsf{sk}' | r' \leftarrow \mathcal{O}(\mathsf{ct}_2)$
4. Compute $a' \leftarrow \mathsf{QFHE.Dec}(\mathsf{sk}', \mathsf{ct}_1)$, $b' \leftarrow \mathsf{QFHE.Dec}(\mathsf{sk}', \mathsf{ct}_2)$.
5. Output $C_{a',b',r',\mathsf{pk}',\mathcal{O}'}$.

We claim that with probability negligibly close to 1, $(a', b', r', \mathsf{pk}', \mathcal{O}') = (a, b, r, \mathsf{pk}, \mathcal{O})$ when $C := C_{a,b,r,\mathsf{pk},\mathcal{O}} \leftarrow \mathcal{D}_{\mathcal{C}}$. This would finish our proof.

Lets analyze the outputs of $\mathcal{B}$ step-by-step.

- After Step (1), with probability neglibibly close to 1, we have that $\mathsf{ct}_1 = \mathsf{QFHE.Enc}(\mathsf{pk}, a; r)$, $\mathsf{pk}' = \mathsf{pk}$, and $\mathcal{O}' = \mathcal{O} \leftarrow \mathsf{LO.Obf}(\mathbf{C}[\mathsf{QFHE.Dec}(\mathsf{sk}, \cdot), b, (\mathsf{sk}|r)])$. Furthermore, we have that ρ' is negligibly close in trace distance to ρ_C.
- Conditioned on Step (1) computing $C(0 \cdots 0)$ correctly, we have that $\mathsf{QFHE.Eval}(U_C(\rho', .), \mathsf{ct}_1)$ computes correctly with probability negligibly close to 1. This is because $\|\rho' - \rho_C\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda)$, and by correctness of both QFHE and (U_C, ρ_C). Conditioned on $\mathsf{ct}_1 = \mathsf{QFHE.Enc}(\mathsf{pk}, a; r)$, when Step (2) evaluates correctly, we have $\mathsf{ct}_2 = \mathsf{QFHE.Enc}(\mathsf{pk}, C(a)) = \mathsf{QFHE.Enc}(\mathsf{pk}, b)$
- Conditioned on $\mathsf{ct}_2 = \mathsf{QFHE.Enc}(\mathsf{pk}, b)$, by correctness of lockable obfuscation, we have that $\mathcal{O}(\mathsf{ct}_2)$ outputs $\mathsf{sk}|r$. Furthermore, by correctness of QFHE, decryption is correct: $\mathsf{QFHE.Dec}(\mathsf{sk}, \mathsf{ct}_1)$ outputs a with probability neglibly close to 1, and $\mathsf{QFHE.Dec}(\mathsf{sk}, \mathsf{ct}_2)$ outputs b with probability neglibly close to 1.

With probability negligibly close to 1, we have shown that $(a', b', r', \mathsf{pk}', \mathcal{O}') = (a, b, r, \mathsf{pk}, \mathcal{O})$.

Note that it is also possible to recover ρ'' that is neglibly close in trace distance to ρ_C. This is because $\sigma = \mathsf{QFHE.Enc}(\mathsf{pk}, \rho'')$ for some ρ'' satisfying $\|\rho'' - \rho_C\|_{\mathrm{tr}}$. Once $\mathsf{sk}' = \mathsf{sk}$ has been recovered, it is possible to also decrypt σ and obtain ρ''. To summarize, we have shown a QPT $\mathcal{B}$ satisfying

$$\mathsf{Pr}[\mathcal{B}(U_C, \rho_C) = (\rho'', C) \ : \ C \leftarrow \mathcal{D}_{\mathcal{C}}] \geq 1 - \mathrm{negl}(\lambda)$$

where $\|\rho'' - \rho_C\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda)$.

Implications to Copy-Protection. We have constructed a class $\mathcal{C}$ and an associated distribution $\mathcal{D}_{\mathcal{C}}$ that is efficient de-quantumizable. In particular, this means that there is no copy-protection for $\mathcal{C}$. If for all inputs x, there is a QPT (U_C, ρ_C) to compute $U_C(\rho_C, x) = C(x)$ with probability $1 - \varepsilon$ for some negligible ε, then

it is possible to find, with probability close to 1, a circuit C' that computes the same functionality as C. We also proved that $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$ is quantum unlearnable. We summarize the result in the following corollary,

Corollary 2. *There is $(\mathcal{C}, \mathcal{D}_{\mathcal{C}})$ that is quantum unlearnable, but $\mathcal{C}$ cannot be copy-protected against $\mathcal{D}_{\mathcal{C}}$. Specifically, for any $C \leftarrow \mathcal{D}_{\mathcal{C}}$ with input length n, and for any QPT algorithm (U_C, ρ_C) satisfying that for all $x \in \{0,1\}^n$,*

$$\mathsf{Pr}[U_C(\rho_C, x) = C(x)] \geq 1 - \varepsilon$$

for some negligible ε, there is a QPT algorithm (pirate) that outputs a circuit C', satisfying $C'(x) = C(x)$ for all $x \in \{0,1\}^n$, with probability negligibly close to 1.

Further Discussion. Notice that in our proof that $\mathcal{C}$ is efficient de-quantumizable, we just need to compute $U_C(\rho_C, x)$ at two different points $x_1 = 0 \cdots 0$ and $x_2 = a$, where the evaluation at x_2 is done homomorphically. This means that any scheme that lets a user evaluate a circuit C at least 2 times (for 2 possibly different inputs) with non-negligible probability cannot be copy-protected. Such a user would be able to find all the parameters of the circuit, $(a, b, r, \mathsf{pk}, \mathcal{O})$, succesfully with non-negligible probability, hence it can prepare as many copies of a functionally equivalent circuit C'.

In our proof, we make use of the fact that (U_C, ρ_C) evaluates correctly with probability close to 1. This is in order to ensure that the pirate can indeed evaluate at 2 points by uncomputing after it computes $C(0 \cdots 0)$. Since any copy-protection scheme can be amplified to have correctness neglibly close to 1 by providing multiple copies of the copy-protected states, our result also rules out copy-protection for non-negligible correctness parameter ε, as long as the correctness of (U_C, ρ_C) can be amplified to neglibily close to 1 by providing $\rho_C^{\otimes k}$ for some $k = \mathrm{poly}(\lambda)$.

Impossibility of Quantum VBB with Single Uncloneable State. Our techniques also rule out the possibility of quantum VBB for classical circuits. In particular, this rules the possibility of quantum VBB for classical circuits with the obfucated circuit being a single uncloneable state, thus resolving an open problem by Alagic and Fefferman [7].

Proposition 5. *Assuming the quantum hardness of learning with errors and assuming that there is a QFHE satisfying the properties described in Theorem 5, there exists a circuit class $\mathcal{C}$ such that any quantum VBB for $\mathcal{C}$ is insecure.*

Proof. We construct a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where every circuit in $\mathcal{C}_\lambda$ is of the form $C_{a,b,r,\mathsf{pk},\mathcal{O}}$ defined in the proof of Theorem 5.

Given any quantum VBB of $C_{a,b,r,\mathsf{pk},\mathcal{O}}$, there exists an adversary $\mathcal{A}$ that recovers b and outputs the first bit of b. The adversary $\mathcal{A}$ follows steps 1–4 of $\mathcal{B}$ defined in the proof of Proposition 4 and then outputs the first bit of b'. In the same proof, we showed that the probability that $b' = b$ is negligibly close to 1 and thus, the probability it outputs the first bit of b is negligibly close to 1.

On the other hand, any QPT simulator Sim with superposition access to $C_{a,b,r,\mathsf{pk},\mathcal{O}}$ can recover b with probability negligibly close to $1/2$. To prove this, we rely upon the proof of Proposition 3 (see full version for details). Suppose T is the number of superposition queries made by Sim to $C_{a,b,r,\mathsf{pk},\mathcal{O}}$. Let $|\psi^0\rangle$ is the initial state of Sim and more generally, let $|\psi^t\rangle$ be the state of Sim after t queries, for $t \leq T$.

We define an alternate QPT simulator Sim' which predicts the first bit of b with probability negligibly close to Sim. Before we describe Sim', we give the necessary preliminary background. Define $|\phi^t\rangle = U_t U_{t-1} \cdots U_1 |\psi^0\rangle$. We proved the following claim.

Claim. $|\langle \phi^t | \psi^t \rangle| = 1 - \delta_t$ for every $t \in [T]$.

Sim' starts with the initial state $|\psi^0\rangle$. It then computes $|\phi^T\rangle$. If U is a unitary matrix Sim applies on $|\psi^T\rangle$ followed by a measurement of a register $\mathbf{D}$ then Sim' also performs U on $|\phi^T\rangle$ followed by a measurement of $\mathbf{D}$. By the above claim, it then follows that the probability that Sim' outputs 1 is negligibly close to the probability that Sim outputs 1. But the probability that Sim' predicts the first bit of b is $1/2$. Thus, the probability that Sim predicts the first bit of b is negligibly close to $1/2$.

4 Main Construction

In this section, we present the main construction of SSL satisfying infinite-term perfect lessor security. We first start by describing the class of circuits of interest.

4.1 Circuit Class of Interest: Evasive Circuits

The circuit class we consider in our construction of SSL is a subclass of evasive circuits. We recall the definition of evasive circuits below.

Evasive Circuits. Informally, a class of circuits is said to be evasive if a circuit drawn from a suitable distribution outputs 1 on a fixed point with negligible probability.

Definition 7 (Evasive Circuits). *A class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, associated with a distribution $\mathcal{D}_\mathcal{C}$, is said to be* **evasive** *if the following holds: for every $\lambda \in \mathbb{N}$, every $x \in \{0,1\}^{\mathrm{poly}(\lambda)}$,*

$$\Pr_{C \leftarrow \mathcal{D}_\mathcal{C}}[C(x) = 1] \leq \mathit{negl}(\lambda),$$

Compute-and-Compare Circuits. The subclass of circuits that we are interested in is called compute-and-compare circuits, denoted by $\mathcal{C}_{\mathsf{cnc}}$. A compute-and-compare circuit is of the following form: $\mathbf{C}[C, \alpha]$, where α is called a lock and C has output length $|\alpha|$, is defined as follows:

$$\mathbf{C}[C,\alpha](x) = \begin{cases} 1, & \text{if } C(x)=\alpha, \\ 0, & \text{otherwise} \end{cases}$$

Multi-bit Compute-and-Compare Circuits. We can correspondingly define the notion of multi-bit compute-and-compare circuits. A multi-bit compute-and-compare circuit is of the following form:

$$\mathbf{C}[C, \alpha, \mathsf{msg}](x) = \begin{cases} \mathsf{msg}, & \text{if } C(x)=\alpha, \\ 0, & \text{otherwise} \end{cases},$$

where msg is a binary string.
We consider two types of distributions as defined by [39].

Definition 8 (Distributions for Compute-and-Compare Circuits). *We consider the following distributions on $\mathcal{C}_{\mathsf{cnc}}$:*

- $\mathcal{D}_{\mathsf{unpred}}(\lambda)$*: For any* $(\mathbf{C}[C, \alpha])$ *along with* aux *sampled from this unpredictable distribution, it holds that* α *is computationally unpredictable given* (C, aux).
- $\mathcal{D}_{\mathsf{pseud}}(\lambda)$*: For any* $\mathbf{C}[C, \alpha]$ *along with* aux *sampled from this distribution, it holds that* $\mathbf{H}_{\mathsf{HILL}}(\alpha|(C, \mathsf{aux})) \geq \lambda^{\epsilon}$*, for some constant* $\epsilon > 0$*, where* $\mathbf{H}_{\mathsf{HILL}}(\cdot)$ *is the HILL entropy [33].*

Note that with respect to the above distributions, the compute-and-compare class of circuits $\mathcal{C}_{\mathsf{cnc}}$ is evasive.

Searchability. For our construction of SSL for $\mathcal{C}$, we crucially use the fact that given a circuit $C \in \mathcal{C}$, we can read off an input x from the description of C such that $C(x) = 1$. We formalize this by defining a search algorithm $\mathcal{S}$ that on input a circuit C outputs an accepting input for C. For many interesting class of functions, there do exist a corresponding efficiently implementable class of circuits associated with a search algorithm $\mathcal{S}$.

Definition 9 (Searchability). *A class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *is said to be* $\mathcal{S}$*-searchable, with respect to a PPT algorithm* $\mathcal{S}$*, if the following holds: on input* C*,* $\mathcal{S}(C)$ *outputs* x *such that* $C(x) = 1$.

Searchable Compute-and-Compare Circuits: Examples. As mentioned in the introduction, there are natural and interesting classes of searchable compute-and-compare circuits. For completeness, we state them again below with additional examples [39].

- Point circuits $C(\alpha, \cdot)$: the circuit $C(\alpha, \cdot)$ is a point circuit if it takes as input x and outputs $C(\alpha, x) = 1$ iff $x = \alpha$. If we define the class of point circuits suitably, we can find α directly from C_α; for instance, α can be the value assigned to the input wires of C.
- Conjunctions with wild cards $C(S, \alpha, \cdot)$: the circuit $C(S, \alpha, \cdot)$ is a conjunction with wild cards if it takes as input x and outputs $C(S, \alpha, x) = 1$ iff $y = \alpha$, where y is such that $y_i = x_i$ for all $i \in S$. Again, if we define this class of circuits suitably, we can find S and α directly from the description of $C(S, \alpha, \cdot)$. Once we find S and α, we can find the accepting input.

- Affine Tester: the circuit $C(\mathbf{A}, \alpha, \cdot)$ is an affine tester, with $\mathbf{A}, \mathbf{y}$ where $\mathbf{A}$ has a non-trivial kernel space, if it takes as input $\mathbf{x}$ and outputs $C(\mathbf{A}, \alpha, \mathbf{x}) = 1$ iff $\mathbf{A} \cdot \mathbf{x} = \alpha$. By reading off $\mathbf{A}$ and α and using Gaussian elimination we can find $\mathbf{x}$ such that $\mathbf{A} \cdot \mathbf{x} = \alpha$.
- Plaintext equality checker $C(\mathsf{sk}, \alpha, \cdot)$: the circuit $C(\mathsf{sk}, \alpha, \cdot)$, with hardwired values decryption key sk associated with a private key encryption scheme, message α, is a plaintext equality checker if it takes as input a ciphertext ct and outputs $C(\mathsf{sk}, \alpha, \mathsf{ct}) = 1$ iff the decryption of ct with respect to sk is α. By reading off α and sk, we can find a ciphertext such that ct is an encryption of α.

Remark 6. We note that both the candidate constructions of copy-protection for point functions by Aaronson [3] use the fact that the accepting point of the point function is known by whoever is generating the copy-protected circuit.

4.2 Ingredients

We describe the main ingredients used in our construction.

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be the class of $\mathcal{S}$-searchable circuits associated with SSL. We denote $s(\lambda) = \mathrm{poly}(\lambda)$ to be the maximum size of all circuits in $\mathcal{C}_\lambda$. And let $\mathcal{D}_\mathcal{C}$ be the distribution associated with $\mathcal{C}$. All the notions below are described in detail in the full version.

Q-Input-Hiding Obfuscators. The notion of q-input-hiding obfuscators states that given an obfuscated circuit, it should be infeasible for a QPT adversary to find an accepting input; that is, an input on which the circuit outputs 1. We denote the q-input-hiding obfuscator scheme to be $\mathsf{qIHO} = (\mathsf{qIHO.Obf}, \mathsf{qIHO.Eval})$ and the class of circuits associated with this scheme is $\mathcal{C}$.

Subspace Hiding Obfuscation. This notion allows for obfuscating a circuit, associated with subspace A, that checks if an input vector belongs to this subspace A or not. In terms of security, we require that the obfuscation of this circuit is indistinguishable from obfuscation of another circuit that tests membership of a larger random (and hidden) subspace containing A. We denote the scheme to be $\mathsf{shO} = (\mathsf{shO.Obf}, \mathsf{shO.Eval})$. The field associated with shO is $\mathbb{Z}_q$ and the dimensions will be clear in the construction.

Q-Simulation-Extractable Non-Interactive Zero-Knowledge (seNIZK) System. This notion is a strengthening of a non-interactive zero-knowledge (NIZK) system. It guarantees the following property: suppose a malicious adversary, after receiving a simulated NIZK proof, produces another proof. Then, there exists an extractor that can extract the underlying witness associated with this proof with probability negligibly close to the probability of acceptance of the proof. We denote the seNIZK proof system to be $\mathsf{qseNIZK} = (\mathsf{CRSGen}, \mathcal{P}, \mathcal{V})$ and we describe the NP relation associated with this system in the construction. We require this scheme to satisfy sub-exponential security. We refer to the full version for an appropriate instantiation.

4.3 Construction

We describe the scheme of SSL below. We encourage the reader to look at the overview of the construction presented in Sect. 1.2 before reading the formal details below.

- $\mathsf{Setup}(1^\lambda)$: Compute $\mathsf{crs} \leftarrow \mathsf{CRSGen}\left(1^{\lambda_1}\right)$, where $\lambda_1 = \lambda + n$ and n is the input length of the circuit. Output crs.

- $\mathsf{Gen}(\mathsf{crs})$: On input common reference string crs, choose a random $\frac{\lambda}{2}$-dimensional subspace $A \subset \mathbb{Z}_q^\lambda$. Set $\mathrm{sk} = A$.

- $\mathsf{Lessor}(\mathrm{sk} = A, C)$: On input secret key sk, circuit $C \in \mathcal{C}_\lambda$, with input length n,
  1. Prepare the state $|A\rangle = \frac{1}{\sqrt{q^{\lambda/2}}} \sum_{a \in A} |a\rangle$.
  2. Compute $\widetilde{C} \leftarrow \mathsf{qIHO.Obf}(C; r_o)$
  3. Compute $\widetilde{g} \leftarrow \mathsf{shO}(A; r_A)$.
  4. Compute $\widetilde{g_\perp} \leftarrow \mathsf{shO}(A^\perp; r_{A^\perp})$.
  5. Let $x = \mathcal{S}(C)$; that is, x is an accepting point of C.
  6. Let L be the NP language defined by the following NP relation.

$$\mathcal{R}_L := \left\{ \left(\left(\widetilde{g}, \widetilde{g_\perp}, \widetilde{C} \right), \ (A, r_o, r_A, r_{A^\perp}, C, x) \right) \ \middle| \ \begin{array}{c} \widetilde{g} = \mathsf{shO}(A; r_A) \\ \widetilde{g_\perp} = \mathsf{shO}(A^\perp; r_{A^\perp}) \\ \widetilde{C} = \mathsf{qIHO.Obf}(C; r_o), \\ C(x) = 1 \end{array} \right\}.$$

     Compute $\pi \leftarrow \mathcal{P}\left(\mathsf{crs}, \left(\widetilde{g}, \widetilde{g_\perp}, \widetilde{C}\right), (A, r_o, r_A, r_{A^\perp}, C, x)\right)$
  7. Output $\rho_C = |\Phi_C\rangle\langle\Phi_C| = \left(|A\rangle\langle A|, \widetilde{g}, \widetilde{g_\perp}, \widetilde{C}, \pi\right)$.

- $\mathsf{Run}(\mathsf{crs}, \rho_C, x)$:
  1. Parse ρ_C as $\left(\rho, \widetilde{g}, \widetilde{g_\perp}, \widetilde{C}, \pi\right)$. In particular, measure the last 4 registers.
     Note: This lets us assume that the input to those registers is just classical, since anyone about to perform Run *might as well measure those registers themselves.*
  2. We denote the operation $\mathsf{shO.Eval}(\widetilde{g}, |x\rangle|y\rangle) = |x\rangle|y \oplus \mathbb{1}_A(x)\rangle$ by $\widetilde{g}[|x\rangle|y\rangle]$, where $\mathbb{1}_A(x)$ is an indicator function that checks membership in A. Compute $\widetilde{g}[\rho \otimes |0\rangle\langle 0|]$ and measure the second register. Let a denote the outcome bit, and let ρ' be the post-measurement state.
  3. As above, we denote the operation $\mathsf{shO.Eval}(\widetilde{g_\perp}, |x\rangle|y\rangle) = |x\rangle|y \oplus \mathbb{1}_A(x)\rangle$ by $\widetilde{g_\perp}[|x\rangle|y\rangle]$. Compute $\widetilde{g_\perp}[\mathsf{FT}\rho'\mathsf{FT}^\dagger \otimes |0\rangle\langle 0|]$ and measure the second register. Let b denote the outcome bit.
     Note: in Step 2 and 3, Run *is projecting ρ onto $|A\rangle\langle A|$ if $a = 1$ and $b = 1$.*
  4. Afterwards, perform the Fourier Transform again on the first register of the post-measurement state, let ρ'' be the resulting state.
  5. Compute $c \leftarrow \mathcal{V}\left(\mathsf{crs}, \left(\widetilde{g}, \widetilde{g_\perp}, \widetilde{C}\right), \pi\right)$

6. If either $a = 0$ or $b = 0$ or $c = 0$, reject and output $\bot$.
7. Compute $y \leftarrow \mathsf{qIHO.Eval}\left(\widetilde{C}, x\right)$.
8. Output $\left(\rho'', \widetilde{g}, \widetilde{g_\bot}, \widetilde{C}, \pi\right)$ and y.

– $\mathsf{Check}(\mathrm{sk} = A, \rho_C)$:
1. Parse ρ_C as $\left(\rho, \widetilde{g}, \widetilde{g_\bot}, \widetilde{C}, \pi\right)$.
2. Perform the measurement $\{|A\rangle\langle A|, I - |A\rangle\langle A|\}$ on ρ. If the measurement outcome corresponds to $|A\rangle\langle A|$, output 1. Otherwise, output 0.

Lemma 1 (Overwhelming probability of perfect correctness). *The above scheme satisfies $\epsilon = negl(\lambda)$ correctness.*

Proof. We first argue that the correctness of Run holds. Since qIHO is perfectly correct, it suffices to show that Run will not output $\bot$. For this to happen, we need to show that $a, b, c = 1$. Since $\widetilde{g} = \mathsf{shO}(A)$, $\widetilde{g_\bot} = \mathsf{shO}(A^\bot)$, and the input state is $|A\rangle\langle A|$, then $a = 1$ and $b = 1$ with probability negligibly close to 1 by correctness of shO. If π is a correct proof, then by perfect correctness of $\mathsf{qseNIZK}$, we have that $\mathsf{Pr}[c = 1] = 1$.

To see that the correctness of Check also holds, note that the leased state is $\rho = |A\rangle\langle A|$, which will pass the check with probability 1.

Lemma 2. *Fix $\beta = \mu(\lambda)$, where $\mu(\lambda)$ is any non-negligible function. Assuming the security of $\mathsf{qIHO}, \mathsf{qseNIZK}$ and shO, the above scheme satisfies $(\beta, \gamma, \mathcal{D}_\mathcal{C})$-infinite-term perfect lessor security, where γ is a negligible function.*

The proof of the above lemma is presented in the full version.

Acknowledgements. We thank Alex Dalzell and Aram Harrow for helpful discussions. During this work, RL was funded by NSF grant CCF-1729369 MIT-CTP/5204.

A Related Work

Quantum Money and Quantum Lightning. Using quantum mechanics to achieve unforgeability has a history that predates quantum computing itself. Wiesner [40] informally introduced the notion of unforgeable quantum money – unclonable quantum states that can also be (either publicly or privately) verified to be valid states. A few constructions [3,4,27,29,34] achieved quantum money with various features and very recently, in a breakthrough work, Zhandry [41] shows how to construct publicly-verifiable quantum money from cryptographic assumptions.

Certifiable Deletion and Unclonable Encryption. Unclonability has also been studied in the context of encryption schemes. The work of Gottesman [31] studies the problem of quantum tamper detection. Alice can use a quantum state to send Bob an encryption of a classical message m with the guarantee that any eavsdropper could not have cloned the ciphertext. In a recent work, Broadbent

and Lord [23] introduced the notion of unclonable encryption. Roughly speaking, an unclonable encryption allows Alice to give Bob and Charlie an encryption of a classical message m, in the form of a quantum state $\sigma(m)$, such that Bob and Charlie cannot 'split' the state among them.

In a follow-up work, Broadbent and Islam [22], construct a one-time use encryption scheme with certifiable deletion. An encryption scheme has certifiable deletion property, if there is an algorithm to check that a ciphertext was deleted.

Quantum Obfuscation. Our proof of the impossibility of SSL is inspired by the proof of Barak et al. [10] on the impossibility of VBB for arbitrary functions. Alagic and Fefferman [7] formalized the notion of program obfuscation via quantum tools, defining quantum virtual black-box obfuscation (qVBB) and quantum indistinguishability obfuscation (qiO), as the natural quantum analogues to the respective classical notions (VBB and iO). They also proved quantum analogues of some of the previous impossibility results from [10], as well as provided quantum cryptographic applications from qVBB and qiO.

Quantum One-Time Programs and One-Time Tokens. Another related primitive is quantum one-time programs. This primitive wasn shown to be impossible by [21]. This rules out the possibility of having a copy-protection scheme where a single copy of the software is consumed by the evaluation procedure. Despite the lack of quantum one-time programs, there are constructions of secure 'one-time' signature tokens in the oracle models [8,13]. A quantum token for signatures is a quantum state that would let anyone in possession of it to sign an arbitrary document, but only once. The token is destroyed in the signing process.

Recent Work on Copy-Protection. While finishing this manuscript, we became aware of very recent work on copy-protection. Aaronson et al. [5] constructed copy-protection for unlearnable functions relative to a classical oracle. Our work complements their results, since we show that obtaining copy-protection in the standard model (i.e., without oracles) is not possible.

References

1. How microsoft corporation makes most of its money. https://www.fool.com/investing/2017/06/29/how-microsoft-corporation-makes-most-of-its-money.aspx
2. Scott Aaronson. Shtetl-Optimized. Ask Me Anything: Apocalypse Edition. https://www.scottaaronson.com/blog/?p=4684#comment-1834174. Comment #283, Posted: 03–24-2020. Accessed 25 Mar 2020
3. Aaronson, S.: Quantum copy-protection and quantum money. In: 2009 24th Annual IEEE Conference on Computational Complexity, pp. 229–242. IEEE (2009)
4. Aaronson, S., Christiano, P.: Quantum money from hidden subspaces. In: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, pp. 41–60 (2012)
5. Aaronson, S., Liu, J., Zhang, R.: Quantum copy-protection from hidden subspaces. arXiv preprint arXiv:2004.09674 (2020)

6. Alagic, G., Brakerski, Z., Dulek, Y., Schaffner, C.: Impossibility of quantum virtual black-box obfuscation of classical circuits. arXiv preprint arXiv:2005.06432 (2020)
7. Alagic, G., Fefferman, B.: On quantum obfuscation. arXiv preprint arXiv:1602.01771 (2016)
8. Amos, R., Georgiou, M., Kiayias, A., Zhandry, M.: One-shot signatures and applications to hybrid quantum/classical authentication. Cryptology ePrint Archive, Report 2020/107 (2020)
9. Ananth, P., La Placa, R.L.: Secure quantum extraction protocols. Cryptology ePrint Archive, Report 2019/1323 (2019)
10. Barak, B.: How to go beyond the black-box simulation barrier. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pp. 106–115. IEEE (2001)
11. Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Lindell, Yehuda (ed.) TCC 2014. LNCS, vol. 8349, pp. 26–51. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_2
12. Barak, B., et al.: On the (Im)possibility of obfuscating programs. In: Kilian, Joe (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
13. Ben-David, S., Sattath, O.: Quantum tokens for digital signatures. arXiv preprint arXiv:1609.09047 (2016)
14. Bitansky, N., Khurana, D., Paneth, O.: Weak zero-knowledge beyond the black-box barrier. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, pp. 1091–1102. ACM (2019)
15. Bitansky, N., Paneth, O.: On the impossibility of approximate obfuscation and applications to resettable cryptography. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 241–250 (2013)
16. Bitansky, N., Shmueli, O.: Post-quantum zero knowledge in constant rounds. In: STOC (2020)
17. Brakerski, Z.: Quantum FHE (Almost) as secure as classical. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 67–95. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_3
18. Brakerski, Z., Döttling, N., Garg, S.: and Giulio Malavolta. Circular-secure lwe suffices, Factoring and pairings are not necessary for io (2020)
19. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 190–213. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_8
20. Broadbent, A., Grilo, A.B.: Zero-knowledge for qma from locally simulatable proofs. arXiv preprint arXiv:1911.07782 (2019)
21. Broadbent, A., Gutoski, G., Stebila, D.: Quantum one-time programs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 344–360. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_20
22. Broadbent, A., Islam, R.: Quantum encryption with certified deletion. arXiv preprint arXiv:1910.03551 (2019)
23. Broadbent, A., Lord, S.: Uncloneable quantum encryption via random oracles. arXiv preprint arXiv:1903.00130 (2019)
24. Coladangelo, A.: Smart contracts meet quantum cryptography. arXiv preprint arXiv:1902.05214 (2019)
25. Coladangelo, A., Vidick, T., Zhang, T.: Non-interactive zero-knowledge arguments for qma, with preprocessing. arXiv preprint arXiv:1911.07546 (2019)

26. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_33
27. Farhi, E., Gosset, D., Hassidim, A., Lutomirski, A., Shor, P.: Quantum money from knots. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 276–289 (2012)
28. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
29. Gavinsky, D.: Quantum money with classical verification. In: 2012 IEEE 27th Conference on Computational Complexity, pp. 42–52. IEEE (2012)
30. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. Technical report, Cryptology ePrint Archive, Report 2020/1010 (2020)
31. Gottesman, D.: Uncloneable encryption. Quant. Inf. Comput. **3**(6), 581–602 (2003)
32. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: FOCS (2017)
33. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28**(4), 1364–1396 (1999)
34. Lutomirski, A., et al.: Breaking and making quantum money: toward a new quantum cryptographic protocol. arXiv preprint arXiv:0912.3825 (2009)
35. Mahadev, U.: Classical homomorphic encryption for quantum circuits. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 332–338. IEEE (2018)
36. Mahadev, U.: Classical verification of quantum computations. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pp. 259–267. IEEE (2018)
37. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039), pp. 543–553. IEEE (1999)
38. Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling (2020)
39. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pp. 600–611. IEEE (2017)
40. Wiesner, S.: Conjugate coding. ACM Sigact News **15**(1), 78–88 (1983)
41. Zhandry, M.: Quantum lightning never strikes the same state twice. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 408–438. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_14