New Approaches for Quantum Copy-Protection

Scott Aaronson[1], Jiahui Liu[1], Qipeng Liu[2], Mark Zhandry[3], and Ruizhe Zhang[1]

[1] The University of Texas at Austin
{aaronson, jiahui, rzzhang}@cs.utexas.edu
[2] Princeton University, USA
qipengl@cs.princeton.edu
[3] Princeton University & NTT Research, USA
mzhandry@princeton.edu

Abstract. Quantum copy-protection uses the unclonability of quantum states to construct quantum software that provably cannot be pirated. copy-protection would be immensely useful, but unfortunately, little is known about achieving it in general. In this work, we make progress on this goal, by giving the following results:

- We show how to copy-protect any program that cannot be learned from its input-output behavior relative to a *classical* oracle. This construction improves on Aaronson (CCC 2009), which achieves the same relative to a quantum oracle. By instantiating the oracle with post-quantum candidate obfuscation schemes, we obtain a heuristic construction of copy-protection.
- We show, roughly, that any program which can be watermarked can be copy *detected*, a weaker version of copy-protection that does not prevent copying, but guarantees that any copying can be detected. Our scheme relies on the security of the assumed watermarking, plus the assumed existence of public-key quantum money. Our construction is publicly detectable and applicable to many recent watermarking schemes and .

1 Introduction

Quantum copy-protection, proposed by Aaronson [Aar09], aims to use the unclonability of quantum states to achieve programs that cannot be copied. That is, the program f is given as a quantum state $|\psi_f\rangle$. $|\psi_f\rangle$ allows for computing f on arbitrary inputs; meanwhile, it is infeasible to copy the state $|\psi_f\rangle$, or even convert $|\psi_f\rangle$ into two arbitrary states that both allow for computing f. The quantum no-cloning theorem shows that quantum states, in general, cannot be copied. Copy protection takes this much further, augmenting the unclonable state with the ability to evaluate programs. Copy-protection would have numerous applications to intellectual property management and to cryptography generally.

Progress on quantum copy-protection has unfortunately been slow. On the negative side, copy-protection for general programs is impossible. As explained

by Aaronson [Aar09], any *learnable* program—that is, a program whose description can be learned from just its input/output behavior—cannot be copy-protected. Indeed, given the (copy-protected) code for the program, an attacker can just query the code on several inputs and learn the original program from the results. The attacker can then copy the original program indefinitely. A more recent result of Ananth and La Placa [AP20] shows, under certain computational assumptions, that certain contrived unlearnable programs cannot be copy-protected.

On the positive side, Aaronson demonstrates a quantum oracle[iv] relative to which copy-protection exists for any unlearnable program. Due to the negative result above, this scheme cannot be instantiated in general. Worse, even for programs that are not subject to the impossibility result, it remains unclear how even heuristically to instantiate the scheme. Very recently, Ananth and La Placa [AP20] build a version of copy-protection, which they call software leasing, which guarantees a sort of copy *detection* mechanism. Unfortunately, their work explicitly allows copying the functionality and only ensures that such copying can be detected. Also, their construction only works for a certain class of "evasive" functions, which accept a hidden sparse set of inputs. The work of Ben-David and Sattath [BDS16] and more recently Amos et al. [AGKZ20] can be seen as copy-protecting specific cryptographic functionalities.

1.1 This Work

In this work, we give new general results for copy-protection. Our two main results are:

- Any unlearnable functionality can be copy-protected, relative to a *classical* oracle.
- Any functionality that can be *watermarked* in a certain sense, can be copy-*detected* assuming just the existence of public-key quantum money.

Both of our results are very general, applying to a wide variety of learning and watermarking settings, including settings where functionality preservation is not required. Along the way to obtaining our results, we give new definitions for security of copy-protection (as well as copy-detection and watermarking), which provide stronger guarantees.

Our first result improves Aaronson [Aar09] to use a classical oracle, which can then heuristically be instantiated using candidate post-quantum obfuscation (e.g. [BGMZ18, BDGM20]), resulting in a concrete candidate copy-protection scheme. Of course, the impossibility of Ananth and La Placa [AP20] means the resulting scheme cannot be secure in the standard model for arbitrary programs. Still, it can be conjectured to be secure for programs not subject to the impossibility.

Our second result complements Ananth and La Placa [AP20]'s positive result for copy-detecting certain evasive functions by copy-detecting arbitrary watermarkable functions. For our purposes, watermarkable functions are those that

[iv] That is, an oracle that implements a quantum operation.

can have a publicly observable "mark" embedded into the program, such that it is infeasible to remove the mark without destroying the functionality. We note that the results (and techniques) are incomparable to [AP20]. First, watermarkable functions are never evasive, so the class of functions considered are disjoint. Second, our security guarantee is much stronger than theirs, which we discuss in Section 1.2.

Taken together, we believe our results strongly suggest that watermarkable functions may be copy-protectable. Concretely, the impossibility result of Ananth and La Placa also applies to copy detection, and our second result shows that watermarkable functions, therefore, circumvent the impossibility. Based on this, we conjecture that our first result, when instantiated with candidate obfuscators, is a secure copy-protection scheme for watermarkable functions. We leave proving or disproving our conjecture as an interesting direction for future work.

1.2 Technical Overview

Definitional Work. We first investigate the definition of quantum copy-protection. We find that existing definitions and other straightforward attempts have several limitations. We therefore carefully develop a strong and general definition of copy-protection to resolve these limitations. In particular, our definition captures attacks where (1) the program is meaningfully copied even if the functionality is technically different, and (2) the program is copied only with a small but detectable probability.

Consider the following attempt of defining quantum copy-protection: we say an adversary successfully pirates a quantum program for computing a function f if it outputs two quantum programs σ_1, σ_2, each of them able to compute f correctly on a large fraction of inputs. Now consider applying this definition to the case where f is a signing algorithm with a particular signing key hard-coded, and suppose there are many valid signatures for each message. Consider a hypothetical adversary who "splits" the program into two pieces, each computing valid signatures; but neither computing the same signature that f produces. Such programs are "good enough" for forging signatures, and the ability to copy a signature-producing program in this way would naturally be considered an attack. However, the usual notions of security for copy-protection do not apply to such programs.

Another example is the copy-protection of public-key encryption. Let f be a decrypting algorithm with a particular decryption key hard-coded. Suppose the split two program pieces only work correctly on a sparse set: they can only decrypt correctly on ciphertexts of m_0, m_1; for ciphertexts of other messages, the output is arbitrary. This splitting attack does not violate the security notion either since both functions produced by the adversary differ from the original program on most inputs. But again, such programs are "good enough" for breaking the semantic security of the encryption scheme, and therefore would reasonably count as an attack.

Similar definitional issues are discussed in the context of watermarking [GKM+19] but have not been explored in the setting of copy-protection. Inspired by the

watermarking case, our solution is to define "compute f correctly" by a general relation. The relation takes some random coins r, the function f (with some additional information about f hard-coded in the circuit); it samples an input and runs the (quantum) program on that classical input; finally, it checks the output of the quantum program, testing (in superposition) if the output z together with f, r is in the relation. As an example, if f is a signing circuit (with the signing key hard-coded), the relation is defined as: use random coins r to generate a random message m, run the quantum program on m and test in superposition if it is a valid signature, by applying the verification algorithm $\mathsf{Ver}(\mathsf{vk}, m, \cdot\,; r)$.

Therefore, we propose a general definition that can capture a broader class of attacks, especially in the context of cryptographic functionalities.

Unfortunately, another uniquely quantum issue arises when trying to formulate a definition. We intuitively want to consider a program to be a valid copy if it computes f correctly on a non-trivial fraction of the domain. Unfortunately, there is no physical way to actually test if a program represented as a quantum state satisfies the property when an algorithm only receives a single copy of the program, especially in game-based security definitions.

Generally, any attempt at assigning a non-trivial property to quantum states (e.g., "valid program" vs. "invalid program") will be physically meaningless. Indeed, given any valid program P_1 and any invalid program P_2 (regardless of the meaning of "valid"), what is the uniform superposition $|P_1\rangle + |P_2\rangle$ of the two programs? Is it valid or invalid? Regardless of which, because the three programs are not orthogonal quantum states, no measurement can determine all three states' validity.

At a more operational level, the classical way to test for correctly computing f is to evaluate the function on all points and report the fraction of inputs where the program computed correctly. Alternatively, one can efficiently *estimate* the fraction of inputs that are computed correctly by simply testing a polynomial number of random points. Regardless, testing involves running the program on multiple points.

In the setting of quantum programs, however, the uncertainty principle implies that the moment one tests the first input, the quantum program state is irreversibly altered, potentially affecting the subsequent evaluations of the program. Thus, the fraction of inputs computed correctly is ever-changing, and simply evaluating the program on several points will not give a meaningful indication of the program's validity at any single point in time.

To illustrate further issues, consider the adversary which takes its quantum program P and simply produces $\frac{1}{\sqrt{2}}(|P\rangle\,|D\rangle\,|0\rangle + |D\rangle\,|P\rangle\,|1\rangle)$ where D is a dummy program that outputs junk. The two halves of this bipartite system each have probability $1/2$ of outputting the right answer on a random input. And yet, this "attack" is rather useless and should not be considered a break.

On the other hand, consider a hypothetical attacker which produces $\frac{1}{\sqrt{2}}\,|P\rangle\,|P\rangle + \frac{1}{\sqrt{2}}\,|D\rangle\,|D\rangle$. The two halves of this bipartite system each separately has probability $1/2$ of outputting the right answer on a random input. However, if we

measure both halves, there is a $1/2$ chance of obtaining two copies of P, which each answers correctly with probability 1. Therefore, this attack should likely be considered a break.

Thus, we see that any characterization of program validity which just tests the program on a random input cannot distinguish cases that should be considered breaks from those that are not. On the other hand, if we test multiple random inputs, we run into the problem that testing each input causes the program state to change, meaning we may not get meaningful results.

Our solution will be to use recent ideas from Zhandry [Zha20], who considered similar issues in the context of traitor tracing. At a high level, the issue above is that we are trying to assign a property to a quantum state (whether the state is a good program), but this property is non-physical and does not make sense for mixed or entangled states. Instead, we want "a program is good" to be a measurement that can be applied to the state. We would naturally also want the measurement to be projective, so that if a program is once tested to be "good", it will always be "good".

Let $\mathcal{M} = (M_0, M_1)$ be binary positive operator-valued measures (POVMs) that represents choosing random coins and testing if the quantum program computes correctly with respect to the random coins. For a mixed quantum program state σ, the probability the program evaluates correctly relative to this test is given as $\mathsf{Tr}[M_0\sigma]$. Let $\mathcal{M}'$ be the (inefficient) projective measurement $\{P_p\}_{p\in[0,1]}$, projecting onto the eigenspaces of M_0, where p ranges over the corresponding eigenvalues of M_0.[v] Zhandry showed that the measurement below results in an equivalent POVM as $\mathcal{M}$:

- Apply the projective measurement $\mathcal{M}'$, and obtain p;
- Output 0 with probability p, and output 1 with probability $1 - p$.

Intuitively, $\mathcal{M}'$ will project a state to an eigenvector with eigenvalue p. The leftover state computes correctly on p-fraction of all inputs.

Therefore, we say a quantum program σ is tested to be γ-good, if the measurement $\mathcal{M}'$ has outcome $p \geq \gamma$. We say an adversary successfully pirates a quantum program for computing f if the two programs are both tested to be γ-good with non-negligible probability. We will show an efficient algorithm that approximates the measurement. Thus, our new definition provides an operational game-based security definition that resolves all the issues we mentioned above. Besides, although the definition may be laborious, this definition implies the previous definitions in [Aar09, CMP20] and etc, and we find proving security with this definition is considerably easier. Using similar ideas, we also define quantum unlearnability of programs and quantum copy-detection.

Our Copy-Protection Scheme. We give a quantum copy-protection construction for all unlearnable functions based on (1) classical oracles, and (2) subspace states, or more abstractly, any tokenized signature scheme [BDS16].

[v] Since $M_0 + M_1$ is the identity, M_1 shares the same eigenvectors, with eigenvalue $1 - p$.

Note the difference between classical and quantum oracles: a classical oracle is a classical functionality that can be (superposition) queried in a black-box way, while a quantum oracle is a quantum unitary operation used as a black-box. It is more feasible to implement classical oracles heuristically considering existing candidates of (post-quantum) obfuscations for classical circuits [BDGM20]. Therefore our construction is a significant improvement from the result in [Aar09].

A tokenized signature generates a signature token $|\mathsf{sig}\rangle$ which we call a signing token. A signer who gets one copy of the signing token can sign a single bit b of her choice. $\mathsf{Sign}(b, |\mathsf{sig}\rangle)$ outputs a classical signature whose correctness guarantee is the same as classical signatures: namely, verification will accept the result as a signature on b. Importantly, the signing procedure is a unitary and will produce a superposition of all valid signatures of b; to obtain a classical signature, a measurement to the state is necessary, leading to a collapse of the token state. Thus, a signature token $|\mathsf{sig}\rangle$ can only be used to produce one classical signature of a single bit, and any attempt to produce a classical signature of the other bit would fail. [BDS16] formalizes this idea and constructs a tokenized signature scheme relative to a classical oracle (a subspace membership oracle).

The high-level idea of our copy-protection scheme is that it requires an authorized user to query an oracle twice on signatures of bits 0 and 1. Let f be the function we want to copy-protect. Define the following classical circuits:

$$\mathcal{O}_1(x, \mathsf{sig}) = \begin{cases} H(x) & \text{if } \mathsf{Ver}(\mathsf{vk}, 0, \mathsf{sig}) = 1 \\ \bot & \text{otherwise} \end{cases},$$

$$\mathcal{O}_2(x, \mathsf{sig}) = \begin{cases} f(x) \oplus H(x) & \text{if } \mathsf{Ver}(\mathsf{vk}, 1, \mathsf{sig}) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

Here H is a random function. The copy-protected program of f is a signature token $|\mathsf{sig}\rangle$ and obfuscations of $\mathcal{O}_1, \mathcal{O}_2$, which we will heuristically treat as oracles to $\mathcal{O}_1, \mathcal{O}_2$. We denote this program as $(|\mathsf{sig}\rangle, \mathcal{O}_1, \mathcal{O}_2)$.

To obtain $f(x)$, a user has to query on signatures of both bits and get $H(x)$ and $H(x) \oplus f(x)$. Note that even if one can only produce one of the classical signatures with token $|\mathsf{sig}\rangle$, a user can still query both oracles $\mathcal{O}_1, \mathcal{O}_2$ multiple times. To obtain $H(x)$, a user can compute the superposition of all valid signatures of 0 by applying a unitary and feed the quantum state together with x to $\mathcal{O}_1$. It then measures the output register. The user never actually measures the signature. Because the output register contains a unique output $H(x)$, by Gentle Measurement Lemma [Aar04], it can rewind the quantum state back to $|\mathsf{sig}\rangle$. Thus, our copy-protection scheme allows a copy-protected program to be evaluated on multiple inputs multiple times.

We next show how to prove anti-piracy security. Let σ_1, σ_2 be two (potentially entangled) program states pirated by an adversary, which makes oracle access to both $\mathcal{O}_1, \mathcal{O}_2$ and breaks the anti-piracy security. Let $\mathcal{O}_\bot$ be an oracle that always outputs $\bot$. If σ_1 never queries the oracle $\mathcal{O}_2$, we know the two programs $(\sigma_1, \mathcal{O}_1, \mathcal{O}_2)$ and $(\sigma_1, \mathcal{O}_1, \mathcal{O}_\bot)$ will have identical output distributions. Moreover,

$(\sigma_1, \mathcal{O}_1, \mathcal{O}_\perp)$ can be simulated even without querying f because $\mathcal{O}_1$ is simply a random oracle (on valid inputs). Therefore, the program can be used to break the unlearnability of f. Similarly, if σ_2 never queries the oracle $\mathcal{O}_1$, the program $(\sigma_2, \mathcal{O}_\perp, \mathcal{O}_2)$ can be used to break the unlearnability of f.

Since f is unlearnable, the above two cases can not happen. We show that under this case, we can extract signatures of both 0 and 1. Intuitively, since $(\sigma_1, \mathcal{O}_1, \mathcal{O}_2)$ makes queries to $\mathcal{O}_2$, we can run the program on random inputs and measure a random query to $\mathcal{O}_2$, thereby extracting a signature of 1. Similarly, it holds for $(\sigma_2, \mathcal{O}_1, \mathcal{O}_2)$ and one could extract a signature of 0. Unfortunately, this intuition does not quite work since σ_1 and σ_2 are potentially entangled. It means there can be correlations between the outcomes of the measurements producing the two signatures: perhaps, if the measurement on $(\sigma_1, \mathcal{O}_1, \mathcal{O}_2)$ produces a valid signature on 1, then the measurement on $(\sigma_2, \mathcal{O}_1, \mathcal{O}_2)$ is guaranteed to fail to produce a signature. We show by a delicate argument that, in fact, adversaries cannot cheat using such correlations. Intuitively, although σ_1, σ_2 are entangled, we show there exists an efficient measurement: by applying this measurement to (σ_1, σ_2), with non-negligible probability, the resulting programs (σ_1', σ_2') have the following properties:

- They are both "good" programs. Thus, we can extract a signature of 1 in σ_1'.
- The resulting program σ_2'' after applying any measurement on σ_1' is still "good". Similarly, we can extract a signature of 0 in σ_2''.

Note that the above argument does not directly work for the original programs (σ_1, σ_2).

Our Copy-Detection Scheme. Inspired by [AP20], we propose a weaker definition called copy-detection, which has an additional checking procedure. A user can publicly verify a program's validity by running this checking procedure. The security guarantees that, given one copy of the program, no adversary can produce two programs such that both programs pass the checking procedure and both are 'functionally correct' (as in the copy-protection definition) — in other words, *honest* users can always identify the pirate. Looking ahead, we note that copy-detection is similar to secure software leasing (SSL, [AP20]), with the major differences are (1) the checking procedure is public, (2) 'functionally correct' in the security of copy-detection is average-case while that in the security of SSL is worst-case.

We construct a copy-detection scheme for any function family that can be watermarked. A watermarking scheme roughly consists of the following procedure: Mark takes a circuit and a message, and outputs a circuit embedded with that mark; $\mathsf{Extract}$ takes a marked circuit and produces the embedded mark. A watermarking scheme requires: (1) the watermarked circuit $\tilde{f} = \mathsf{Mark}(f, m)$ should preserve its intended functionality as f; (2) any efficient adversary given a marked $\tilde{f}$, can not generate a new marked circuit with a different mark (or remove the mark) while preserving its functionality. Watermarking primitives have

been studied in previous works, including [CHN+18, KW17, QWZ18, KW19, GKM+19].

Our construction also requires a public key quantum money scheme. It consists two procedures: a generation procedure and a verification procedure. The generation procedure outputs a quantum banknote $|\$\rangle$. Verification is public, takes a quantum money banknote, and outputs either a (classical) serial number of that banknote or $\perp$ indicating it is an invalid banknote. The security requires no efficient adversary could use a quantum banknote $|\$\rangle$ to prepare two quantum banknotes $|\$_1\rangle |\$_2\rangle$ such that both banknotes pass the verification and their serial numbers are equal to that of $|\$\rangle$. We note that this version of quantum money corresponds to a "mini-scheme" as defined by [AC12].

The copy-detection scheme takes a function f, samples a banknote $|\$\rangle$ with serial number s, lets $\tilde{f} \leftarrow \mathsf{Mark}(f, s)$ and outputs a copy-detection program as $(\tilde{f}, |\$\rangle)$. To evaluate the function, it simply runs the classical program $\tilde{f}$. To check a program is valid, it extracts the serial number from the money state and compares it with the mark of the program.

The security requires that no efficient adversary could produce $\tilde{f}_1, |\$_1\rangle$ and $\tilde{f}_2, |\$_2\rangle$ such that two programs pass the check and both classical circuits preserve the functionality. Let s be the serial number of $|\$\rangle$, s_b be the serial number of $|\$_b\rangle$ for $b = 1, 2$. To pass the check, there are two possible cases:

- $s_1 = s_2 = s$. In this case, $|\$_1\rangle |\$_2\rangle$ breaks the security of the quantum money scheme because one successfully duplicates a banknote with the same serial number.
- At least one of $s_b \neq s$. Because the mark of $\tilde{f}_b$ is also equal to s_b, one of $\tilde{f}_b$ breaks the security of the watermarking scheme, as it preserves the functionality, while having a different mark from s.

We show that the above construction and proof apply to a wide range of watermarking primitives.

Copy-Protection in the Standard Model? The security of our copy-protection scheme requires treating the obfuscated programs as oracles. While we prove security for all unlearnable programs, we cannot expect such security to hold in the standard model: as shown in [AP20], there are unlearnable functions that can not be copy-protected or even copy-detected. On the other hand, watermarkable programs are a natural class of programs that are necessarily immune to the style of counter-example of Barak et al. [BGI+01], on which the copy-protection impossibility is based. Namely, the counter-example works by giving programs that are unlearnable, but such that having any (even approximate [BP15]) code for the program lets you recover the original program. Such programs *cannot* be watermarkable, as the adversary can always recover the original program from the (supposedly) watermarked program.

Thus, we broadly conjecture that all watermarkable functions can be copy-protected. Our copy-detection result gives some evidence that this may be feasible. Concretely, we conjecture that our copy-protection construction is secure

for any watermarkable program when the oracles are instantiated with post-quantum obfuscation constructions. We leave justifying either the broad or concrete conjectures as fascinating open questions.

1.3 Other Related Works

Quantum Copy Protection Quantum copy-protection was proposed by Aaronson in [Aar09]; this paper gave two candidate schemes for copy-protecting point functions without security proofs and showed that any functions that are not quantum learnable could be quantum copy-protected relative to a quantum oracle (an oracle which could perform an arbitrary unitary).

[AP20] gave a conditional impossibility of general copy-protection: they construct a quantum unlearnable circuit using the quantum FHE scheme and compute-and-compare obfuscation [WZ17, GKW17], which is not copy-protectable once a QPT adversary has non-black-box access to the program. [AP20] also gave a new definition that is weaker than the standard copy-protection security, called Secure Software Leasing (SSL) and an SSL construction for a subclass of evasive functions, namely, searchable compute-and-compare circuits.

[BL19] and [GZ20] introduced two new notions respectively, unclonable encryption/ decryption schemes; [CMP20] gave a construction for copy-protecting point functions in the quantum random oracle model with techniques inspired by [BL19] and the construction can be extended to copy-protecting compute-and-compare circuits. [BJL+21] then constructed information-theoretic SSL for point functions.

1.4 Concurrent and Independent Work

Very recently, [KNY20] presents a secure software leasing for a subclass of evasive functions and PRFs, using watermarking and two-tier quantum-lightning, which can be built from the LWE assumption. Their main observation is that the full power of public-key quantum money is not needed in the verification of SSL, and they introduce a new primitive in between public-key and private-key quantum money, which they call two-tier quantum lightning. While their construction can be built from LWE alone, our construction aims at a more generalized definition in terms of successful piracy and functionality-preserving; our copy detection construction also works for a broader class of cryptographic functionalities such as encryption and signature.

2 Preliminaries

We denote by λ the security parameter, and when inputted into an algorithm, λ will be represented in unary. We say a function $\epsilon(x)$ is *negligible* if for all inverse polynomials $1/p(x)$, $\epsilon(x) < 1/p(x)$ for all large enough x. We denote a negligible function by $\mathsf{negl}(x)$. We use QPT to denote quantum polynomial time.

2.1 Quantum Computation

Definition 1 (Trace distance). *Let $\rho, \sigma \in \mathbb{C}^{2^n \times 2^n}$ be the density matrices of two quantum states. The trace distance between ρ and σ is*

$$\|\rho - \sigma\|_{\mathrm{tr}} := \frac{1}{2}\sqrt{\mathrm{Tr}[(\rho-\sigma)^\dagger(\rho-\sigma)]},$$

Here, we only state a key lemma for our construction: the Gentle Measurement Lemma proposed by Aaronson [Aar04], which gives a way to perform measurements without destroying the state.

Lemma 1 (Gentle Measurement Lemma [Aar04]). *Suppose a measurement on a mixed state ρ yields a particular outcome with probability $1-\epsilon$. Then after the measurement, one can recover a state $\tilde{\rho}$ such that $\|\tilde{\rho} - \rho\|_{\mathrm{tr}} \leq \sqrt{\epsilon}$. Here $\|\cdot\|_{\mathrm{tr}}$ is the trace distance (defined in Definition 1).*

We give other basic definitions of quantum computation and quantum information in the full version.

2.2 Quantum Oracle Algorithm

We consider the quantum query model in this work, which gives quantum circuits access to some oracles.

Definition 2 (Classical Oracle). *A classical oracle $\mathcal{O}$ on input query x is a unitary transformation of the form $U_f|x,y,z\rangle \to |x,y+f(x),z\rangle$ for classical function $f : \{0,1\}^n \to \{0,1\}^m$. Note that a classical oracle can be queried in quantum superposition.*

In the rest of the paper, we refer to the word "oracle" as "classical oracle". A quantum oracle algorithm with oracle access to $\mathcal{O}$ is a sequence of unitary U_i and oracle access to $\mathcal{O}$ (or U_f). Thus, the query complexity of a quantum oracle algorithm is defined as the number of $\mathcal{O}$ access.

In the analysis of the security of the copy-protection scheme in Section 5.2, we will use the following theorem from [BBBV97] to bound the change in adversary's state when we change the oracle's input-output behavior at places where the adversary hardly ever queries on.

Theorem 1 ([BBBV97]). *Let $|\phi_i\rangle$ be the superposition of quantum Turing machine $\mathcal{M}$ with oracle $\mathcal{O}$ on input x at time i. Define $W_y(|\phi_i\rangle)$ to be the sum of squared magnitudes in $|\phi_i\rangle$ of configurations of $\mathcal{M}$ which are querying the oracle on string y. For $\epsilon > 0$, let $F \subseteq [0, T-1] \times \Sigma^*$ be the set of time-string pairs such that $\sum_{(i,y)\in F} W_y(|\phi_i\rangle) \leq \epsilon^2/T$.*

Now suppose the answer to each query $(i,y) \in F$ is modified to some arbitrary fixed $a_{i,y}$ (these answers need not be consistent with an oracle). Let $|\phi'_i\rangle$ be the superposition of $\mathcal{M}$ on input x at time i with oracle $\mathcal{O}$ modified as stated above. Then $\||\phi_T\rangle - |\phi'_T\rangle\|_{\mathrm{tr}} \leq \epsilon$.

2.3 Direct-Product Problem and Quantum Signature Tokens

In this section, we will define direct-product problems, which are key components of the quantum signature token scheme by Ben-David and Sattath [BDS16] and also our quantum copy-protection scheme.

Definition 3 (Dual Subspace). *Given a subspace S of a vector space V, let $S^\perp$ be the orthogonal complement of S: the set of $y \in V$ such that $x \cdot y = 0$ for all $x \in S$. It is not hard to show: $S^\perp$ is also a subspace of V; $(S^\perp)^\perp = S$.*

Definition 4 (Subspace Membership Oracles). *A subspace membership oracle for a subspace $A \subseteq \mathbb{F}^n$, denoted as U_A, on input vector v, will output 1 if $v \in A$, $v \neq 0$ and output 0 otherwise.*

Definition 5 (Subspace State). *For a subspace $A \subseteq \mathbb{F}^n$, the state $|A\rangle$ is defined as $\frac{1}{\sqrt{|A|}} \sum_{v \in A} |v\rangle$, which is a uniform superposition of all vectors in A.*

Direct-Product Problem Our construction relies on the following problem called the "Direct-Product Problem" in [AC12]: for any QPT adversary $\mathcal{A}$, given one copy of $|A\rangle$ and oracle access to $U_A, U_{A^\perp}$, the problem is to finds two *non-zero* vectors such that $u \in A$ and $v \in A^\perp$.

Ben-David and Sattath [BDS16] proved the hardness of the direct-product problem for the construction of quantum signature tokens. More precisely, a signature token is a subspace state $|A\rangle$ in their construction. All vectors in $A \setminus \{0\}$ are signatures for bit 0 and all vectors in $A^\perp \setminus \{0\}$ are signatures for bit 1. Therefore, to generate valid signatures for both 0 and 1, it is required to solve the "Direct-Product Problem". We believe that our copy-protection scheme works for general signature token schemes. To keep the statements and proofs simple, we focus on the construction in [BDS16].

Theorem 2 ([BDS16]). *Let $\epsilon > 0$ be such that $1/\epsilon = o(2^{n/2})$. Let A be a random subspace $\mathbb{F}^n$, and $\dim(A) = n/2$. Given one copy of $|A\rangle$ and access to both subspace membership oracles of U_A and $U_{A^\perp}$, an adversary needs $\Omega(\sqrt{\epsilon}2^{n/4})$ queries to output a pair of non-zero vectors (u, v) such that $u \in A$ and $v \in A^\perp$ with probability at least ϵ.*

We will refer to the direct-product problem as a security game, which is defined as follows:

Definition 6 (Direct-Product Game). *A direct-product game consists of the following steps:*

- ***Setup Phase**: the challenger takes in a security parameter λ, samples a random $\lambda/2$-dimensional subspace A from $\mathbb{F}^\lambda$; then prepares the membership oracle U_A for A, $U_{A^\perp}$ for the dual subspace $A^\perp$ and a quantum state $|A\rangle$.*
- ***Query Phase**: the challenger sends $|A\rangle$ to the adversary; the adversary can query $U_A, U_{A^\perp}$ for polynomially many times.*
- ***Output Phase**: the adversary outputs two vectors (u, v).*

The challenger checks if $u \in A \setminus \{0\}, v \in A^{\perp} \setminus \{0\}$. *If this is satisfied, then the adversary wins the game.*

Theorem 2 shows that for any QPT adversary, the winning probability of the direct-product game is negligible.

2.4 Testing Quantum Programs: Measurement Implementation

In classical cryptographic security games, the challenger typically gets some information from the adversary and checks if this information satisfies certain properties.

However, in the quantum world, when a challenger tries to decide if a quantum adversary has produced a state with certain properties, especially in the context of security games for properties related to unclonability, classical definitions of "testing properties" may result in various failures as discussed in [Zha20]. Such an issue has also been discussed in the introduction.

To deal with this issue, [Zha20] formalized a measurement procedure for testing an adversary's state. This is best understood with an example.

Consider a security game where the adversary needs to produce a state that can decrypt a challenge ciphertext. First, the challenger's behavior is abstracted into the following:

- Encrypt a random message bit m to get c using randomness rand, note that randomness rand is used to sample m and random coins for encryption;
- Run the adversary's state on the resulting ciphertext c;
- Output 1 or 0 depending on whether the adversary's state correctly decrypts or not.

Fixing the ciphertext c, the procedure of outputting 1 or 0 depending on whether the adversary's state correctly decrypts c can be described as a projective measurement $\mathcal{P}_{m,c} = (P_{m,c}, Q_{m,c})$ where $P_{m,c}$ corresponds to output 1, $Q_{m,c}$ corresponds to output 0 and $(P_{m,c}, Q_{m,c})$ can be efficiently implemented given subscript m, c. The challenger uses m, c as a control to decide which projective measurement to be applied to the state.

More generally, let $\mathcal{R}$ be the set of randomness, $\mathcal{I}$ be the control set (similar to the role of m, c in the above example). Let D be a function from $\mathcal{R}$ to $\mathcal{I}$. For a uniform randomness rand, $D(\mathsf{rand})$ defines a distribution over $\mathcal{I}$. Therefore we will use the word "distribution" for D in the rest of the discussion. For every control (or index) $i \in \mathcal{I}$, we have a projective measurement $\mathcal{P}_i = (P_i, Q_i)$. Let $\mathcal{P} = \{\mathcal{P}_i = (P_i, Q_i)\}$ be a collection of binary projective measurements. We define a mixture of projective measurement $\mathcal{P}_D$ as follows.

Definition 7 (Mixture of Projective Measurement $\mathcal{P}_D$). *For $\mathcal{P} = \{P_i, Q_i\}_{i \in \mathcal{I}}$ and $D : \mathcal{R} \to \mathcal{I}$, a mixture of projective measurement $\mathcal{P}_D = (P_D, Q_D)$ is a POVM defined as the following:*

$$P_D = \sum_{i \in \mathcal{I}} \Pr[i \leftarrow D(R)]\, P_i, \qquad Q_D = \sum_{i \in \mathcal{I}} \Pr[i \leftarrow D(R)]\, Q_i,$$

where R is a uniform random variable in $\mathcal{R}$.

In other words, $\mathcal{P}_D$ is implemented in the following way: sample randomness $\mathsf{rand} \leftarrow \mathcal{R}$, compute index/control $i \leftarrow D(\mathsf{rand})$ and apply projective measurement $\mathcal{P}_i = (P_i, Q_i)$.

Thus, for any quantum state ρ, $\mathrm{Tr}[P_D \rho]$ is the probability that a random sampled projective measurement $\mathcal{P}_i = (P_i, Q_i)$ (according to the distribution D) applies on ρ and outputs 1.

Definition 8 (Projective Implementation). *Let $\mathcal{P} = (P, Q)$ be a binary outcome POVM. Let $\mathcal{D}$ be a finite set of distributions $(p, 1-p)$ over outcomes $\{0,1\}$. Let $\mathcal{E} = \{E_p\}_{(p,1-p)\in\mathcal{D}}$ be a projective measurement with index set $\mathcal{D}$. Consider the following measurement experiment:*

- *Measure under the projective measurement $\mathcal{E}$ and obtain a distribution $(p, 1-p)$ over $\{0,1\}$;*
- *Output a bit according to the distribution: output 1 with probability p and output 0 with probability $1-p$.*

We say the measurement $\mathcal{E}$ is a projective implementation of $\mathcal{P}$ if the above experiment and $\mathcal{P}$ produce identical outcomes on any quantum states. We denote $\mathcal{E}$ by $\mathsf{ProjImp}(\mathcal{P})$.

Note that if the collapsed state is an eigenvector of P corresponding to eigenvalue p, then it is also an eigenvector of Q corresponding to eigenvalue $1-p$.

Lemma 2 (A variation of Lemma 1 in [Zha20]). *Any binary outcome POVM $\mathcal{P} = (P, Q)$ has a unique projective measurement $\mathsf{ProjImp}(\mathcal{P})$.*

In this work, we propose the following new definition corresponding to $\mathsf{ProjImp}$.

Definition 9 (Threshold Implementation). *A threshold implementation with parameter γ of a binary POVM $\mathcal{P} = (P, Q)$ is a variant of projective implementation $\mathsf{ProjImp}(\mathcal{P})$, denoted as $(\mathsf{TI}_\gamma(\mathcal{P}), \mathbf{I} - \mathsf{TI}_\gamma(\mathcal{P}))$:*

- *Measure under the projective measurement $\mathcal{E}$ ($\mathsf{ProjImp}(\mathcal{P})$) and obtain a distribution $(p, 1-p)$ over $\{0,1\}$;*
- *Output a bit according to the distribution $(p, 1-p)$: output 1 if $p \geq \gamma$, or 0 otherwise.*

Remark 1. For any quantum state ρ, the threshold implementation outputs 1 with probability $\mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P})\rho]$, and 0 with probability $1 - \mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P})\rho]$.

Remark 2. For a binary outcome measurement $\mathcal{P} = (P, Q)$, we usually say "perform measurement P on ρ" if $\mathcal{P}$ was performed on ρ. For example, if we say a threshold implementation $\mathsf{TI}(\mathcal{P}_D)$ on a quantum state ρ outputs 1, we refer to apply $(\mathsf{TI}_\gamma(\mathcal{P}_D), \mathbf{I} - \mathsf{TI}_\gamma\mathcal{P}_D)$ on ρ and the outcome is 1.

Approximating Projective Implementation Before describing the theorem of the approximation algorithm, we give two definitions that characterize how good an approximation projective implementation is, which were first introduced in [Zha20].

Definition 10 (Shift Distance). *For two distributions D_0, D_1, the shift distance with parameter ϵ is defined as $\Delta^{\epsilon}_{\mathsf{Shift}}(D_0, D_1)$, which is the smallest quantity δ such that for all $x \in \mathbb{R}$:*

$$\Pr[D_0 \leq x] \leq \Pr[D_1 \leq x + \epsilon] + \delta,$$
$$\Pr[D_1 \leq x] \leq \Pr[D_0 \leq x + \epsilon] + \delta.$$

For two real-valued measurements $\mathcal{M}$ and $\mathcal{N}$ over the same quantum system, the shift distance between $\mathcal{M}$ and $\mathcal{N}$ with parameter ϵ is defined as,

$$\Delta^{\epsilon}_{\mathsf{Shift}}(\mathcal{M}, \mathcal{N}) := \sup_{|\psi\rangle} \Delta^{\epsilon}_{\mathsf{Shift}}\left(\mathcal{M}(|\psi\rangle), \mathcal{N}(|\psi\rangle)\right).$$

Definition 11 ((ϵ, δ)-Almost Projective). *A real-valued quantum measurement $\mathcal{M}$ is said to be (ϵ, δ)-almost projective if for all quantum state $|\psi\rangle$, apply $\mathcal{M}$ twice in a row to $|\psi\rangle$, obtaining outcomes X and Y. Then we have $\Pr[|X - Y| \leq \epsilon] \geq 1 - \delta$.*

The following theorem gives a way to approximate any projective implementation:

Theorem 3 (Theorem 2 in [Zha20]). *Let D be any probability distribution over some control set $\mathcal{I}$ and $\mathcal{P}$ be a collection of projective measurements. For any $0 < \epsilon, \delta < 1$, there exists an algorithm of measurement $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ that satisfies the followings:*

- $\Delta^{\epsilon}_{\mathsf{Shift}}(\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \leq \delta$.
- $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ *is (ϵ, δ)-almost projective.*
- *The expected running time of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is $T_{\mathcal{P},D} \cdot \mathsf{poly}(1/\epsilon, \log(1/\delta))$ where $T_{\mathcal{P},D}$ is the combined running time of D, the procedure mapping i to (P_i, Q_i) and the run-time of measurement (P_i, Q_i).*

3 Learning Game Definitions

In this section, we define unlearnability, copy-protection, copy-detection, and watermarking with respect to a function family and a testing distribution.

We assume a function f is sampled uniformly at random from a function family $\mathcal{F}_\lambda$. To test the correctness of a quantum program (for computing f), it samples an input x from a testing distribution D_λ, runs the quantum program on x, and checks if the output is $f(x)$.

We will give the generalized definitions (for unlearnability, copy-protection, copy-detection, and watermarking) in the full version, which allow for more general sampling procedures and functionality testing. Since our constructions naturally extend to these settings, we leave all the discussions about definitions and proofs in the full version of the paper.

Definition 12 (Quantum Program with Classical Inputs and Outputs). *A quantum program with classical inputs is a pair of quantum state ρ and unitaries $\{U_x\}_{x\in[N]}$ (where $[N]$ is the domain), such that the state of the program evaluated on input x is equal to $U_x\rho U_x^\dagger$. To obtain an output, it measures the first register of $U_x\rho U_x^\dagger$. Moreover, $\{U_x\}_{x\in[N]}$ has a compact classical description which means "applying U_x" can be efficiently computed given x.*

Notation-wise, the input and output space N, M are functions of λ.

3.1 Unlearnability

First, we define γ-goodness testing with respect to a fixed function f and a testing distribution D (over inputs).

Definition 13 (γ-Goodness Test with respect to f, D). *Let $(\rho, \{U_x\}_{x\in[N]})$ be a quantum program for computing a classical function $f : [N] \to [M]$. Let D be a testing distribution over the input space $[N]$.*

- *Define $\mathcal{P}_x = (P_x, Q_x)$ be the following projective measurement:*
  - *On input x, it runs U_x on the quantum state ρ;*
  - *It measures whether the output register is equal to $f(x)$; output 1 if yes, and 0 otherwise.*

  Let $\mathcal{P} = \{\mathcal{P}_x\}$ be a collection of projective measurements.
- *D is the distribution that samples an input: given randomness rand, output $x = D(\mathsf{rand})$.*
- *Let $\mathcal{P}_D = (P_D, Q_D)$ be the mixture of projective measurement defined in Definition 7.*
- *We say a quantum program is tested γ-**good** with respect to f, D, if the threshold implementation $\mathsf{TI}_\gamma(\mathcal{P}_D)$ outputs 1.*

We then define a learning game for a function family $\mathcal{F}$ and a set of testing distribution $\mathcal{D}$. Note that we assume for a fixed security parameter λ, f is drawn *uniformly at random* from $\mathcal{F}_\lambda$ and the testing distribution D_f is efficiently sampleable given the description f.

Definition 14 (Learning Game for $\mathcal{F}, \mathcal{D}$). *A learning game for a function family $\mathcal{F} = \{\mathcal{F}_\lambda : [N] \to [M]\}$, a distribution family $\mathcal{D} = \{D_f\}$, a threshold $\gamma \in (0, 1)$, and an adversary $\mathcal{A}$ is denoted as $\mathsf{LearningGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists of the following steps:*

1. ***Sampling Phase**: At the beginning of the game, the challenger takes a security parameter λ and samples a function $f \leftarrow \mathcal{F}_\lambda$ uniformly at random;*
2. ***Query Phase**: $\mathcal{A}$ then gets oracle access to f;*
3. ***Output Phase**: Finally, $\mathcal{A}$ outputs a quantum program $(\rho, \{U_x\}_{x\in[N]})$.*

The game outputs 1 if and only if the program is tested γ-good with respect to f, D_f.

Definition 15 (Quantum Unlearnability of $\mathcal{F}$ with Testing Distribution $\mathcal{D}$). *A family of functions $\mathcal{F}$ with respect to $\mathcal{D}$ is called γ quantum unlearnable if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all λ:*

$$\Pr\left[b = 1,\, b \leftarrow \mathsf{LearningGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)$$

3.2 Copy-Protection

Definition 16 (Quantum Copy-Protection). *A quantum copy-protection scheme for $\mathcal{F}, \mathcal{D}$ consists of the following procedures:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$*: the setup algorithm takes in a security parameter λ in unary and generates a secret key* sk*.*
- $\mathsf{Generate}(\mathsf{sk}, f) \to (\rho_f, \{U_{f,x}\}_{x\in[N]})$*: on input $f \in \mathcal{F}_\lambda$ and secret key* sk*, the vendor generates a quantum program $(\rho_f, \{U_{f,x}\}_{x\in[N]})$.*
- $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x) \to y$*: given a quantum program, a user can compute the function $f(x)$ on input x by applying $U_{f,x}$ on ρ_f and measuring the first register of the state.*

Efficiency: Setup, $\mathsf{Compute}$ and $\mathsf{Generate}$ should run in $\mathsf{poly}(\lambda)$ time.
Correctness: There exists a negligible function $\mathsf{negl}(\cdot)$ such that: all $\lambda \in \mathbb{N}$, every $f \in \mathcal{F}_\lambda$, all $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, for all $x \in [N]$, apply $U_{f,x}$ on ρ_f and measure the first register, with probability at least $1 - \mathsf{negl}(\lambda)$, the output is a fixed value $z_{f,x}$; moreover, $z_{f,x} = f(x)$.
Security: The γ-anti-piracy security defined below.

Note that correctness ensures that the copy-protected program can be evaluated polynomially many times.

Definition 17 (γ-Anti-Piracy Security Game). *An anti-piracy security game for $\mathcal{F}, \mathcal{D}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists of the following steps:*

1. ***Setup Phase**: At the beginning of the game, the challenger takes a security parameter λ and obtains a secret key $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. ***Sampling Phase**: A function f is sampled uniformly at random, $f \leftarrow \mathcal{F}_\lambda$.*
3. ***Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a copy-protection program for f: $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$.*
4. ***Output Phase**: Finally, $\mathcal{A}$ outputs a (possibly mixed and entangled) state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_x, \{U_{R_2,x}\}_x)$ They can be viewed as programs $\mathsf{P}_1 = (\sigma[R_1], \{U_{R_1,x}\}_{x\in[N]})$ and $\mathsf{P}_2 = (\sigma[R_2], \{U_{R_2,x}\}_{x\in[N]})$.*

The game outputs 1 if and only if both programs $\mathsf{P}_1, \mathsf{P}_2$ are tested to be γ-good with respect to $f, \mathcal{D}_f$.

Definition 18 (γ-Anti-Piracy-Security). *A copy-protection scheme for $\mathcal{F}$ and $\mathcal{D}$ has γ-anti-piracy security, if for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for all $\lambda \in \mathbb{N}$:*

$$\Pr\left[b = 1, b \leftarrow \mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{1}$$

In the full version, we will show our new anti-piracy security implies previous security, defined in [Aar09].

3.3 Copy-Detection

A copy-detection scheme is very similar to the copy-protection scheme, except it has an additional procedure Check which applies a projective measurement and checks if the quantum state is valid.

Definition 19 (Quantum Copy-Detection). *A quantum copy-detection scheme for $\mathcal{F}, \mathcal{D}$ consists of the following procedures:*

$\mathsf{Setup}(1^\lambda)$, $\mathsf{Generate}(\mathsf{sk}, f)$ *and* $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x)$ *are the same as those in Definition 16, except* Setup *additionally samples a public key for* Check.

$\mathsf{Check}(\mathsf{pk}, \rho_f, \{U_{f,x}\}_{x\in[N]}) \to b, \rho'$*: on input a quantum program, it applies a binary projective measurement (P_0, P_1) on ρ_f that depends on $\{U_{f,x}\}_{x\in[N]}$; it outputs the outcome b and the leftover state ρ'.*

Correctness ($\mathsf{Generate}$): The same as the security of Definition 16.

Correctness (Check): For every efficient $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, all $\lambda \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, every $f \in \mathcal{F}_\lambda$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, $\mathsf{Check}(\mathsf{pk}, \rho_f, \{U_{f,x}\}_{x\in[N]})$ outputs 1 with probability at least $1 - \mathsf{negl}(\lambda)$.

Security: The γ-copy-detection security defined below.

Definition 20 (γ-Copy-Detection Security Game). *A copy-detection security game for $\mathcal{F}, \mathcal{D}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists of the following steps:*

1. ***Setup Phase**: At the beginning of the game, the challenger takes a security parameter λ and obtains keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. ***Sampling Phase**: A function f is sampled uniformly at random, $f \leftarrow \mathcal{F}_\lambda$.*
3. ***Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a quantum program for f: $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$.*
4. ***Output Phase**: Finally, $\mathcal{A}$ outputs a (possibly mixed and entangled) state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_x, \{U_{R_2,x}\}_x)$ They can be viewed as programs $P_1 = (\sigma[R_1], \{U_{R_1,x}\}_{x\in[N]})$ and $P_2 = (\sigma[R_2], \{U_{R_2,x}\}_{x\in[N]})$.*

The game outputs 1 *if and only if*

– *Apply* Check *on* P_i *respectively and both outcomes are* 1. *Let* P_i' *be the collapsed program conditioned on outcomes are* 1.
– Both *programs* $\mathsf{P}_1', \mathsf{P}_2'$ *are tested* γ*-good with respect to* f, D_f.

Definition 21 (γ-Copy-Detection-Security). *A copy-detection scheme for* $\mathcal{F}, \mathcal{D}$ *has* γ*-copy-detection security, if for any QPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[b = 1, b \leftarrow \mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{2}$$

3.4 Watermarking Primitives with Public Extraction

In this section, we formalize watermarking. We will give the generalized notations in the full version.

Definition 22 (Watermarking Primitives for $\mathcal{F}, \mathcal{D}$). *A watermarking scheme for* $\mathcal{F}, \mathcal{D}$ *consists of the following classical algorithms:*

$\mathsf{Setup}(1^\lambda)$*: it takes as input a security parameter* 1^λ *and outputs keys* $(\mathsf{xk}, \mathsf{mk})$*.* xk *is the extracting key and* mk *is the marking key. We only consider the publicly extractable watermarking schemes. Thus,* xk *is always public.*

$\mathsf{Mark}(\mathsf{mk}, f, \tau)$*: it takes a circuit* f *and a message* $\tau \in \mathcal{M}_\lambda$*, outputs a marked circuit* $\widetilde{f}$*.*

$\mathsf{Extract}(\mathsf{xk}, f')$*: it takes a circuit* f' *and outputs a message in* $\{\bot\} \cup \mathcal{M}_\lambda$*.*

It satisfies the following properties.

Definition 23 (Correctness of Mark (Functionality Preserving)). *For for every efficient algorithm* $\mathcal{A}$*, there exists a negligible function* negl*, for all* $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, and every* $\tau \in \mathcal{M}_\lambda$*, all* λ*,*

$$\Pr\left[\tilde{f}(x) = f(x) \ : \ \begin{matrix} f \leftarrow \mathcal{F}_\lambda \\ \tilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau) \\ x \leftarrow D_f \end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 24 (Correctness of Extract). *For every efficient algorithm* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$*, for all* $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, and every* $\tau \in \mathcal{M}_\lambda$*, all* λ*,*

$$\Pr\left[\tau \neq \mathsf{Extract}(\mathsf{xk}, \widetilde{f}) \ : \ \begin{matrix} f \leftarrow \mathcal{F}_\lambda \\ \widetilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau) \end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Definition 25 (γ-Unremovability with respect to $\mathcal{F}, \mathcal{D}$). *Consider the following game, denoted as* $\mathsf{WaterMarkingGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}$*:*

1. ***Setup****: The challenger samples* $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$*.* $\mathcal{A}$ *then gets* xk*.*
2. ***Sampling Phase****: A function* f *is sampled uniformly at random in* $\mathcal{F}_\lambda$*.*
3. ***Query Phase****:* $\mathcal{A}$ *has classical access to* $\mathsf{Mark}(\mathsf{mk}, f, \cdot)$ *at any time. Define* Q *be the set of messages that* $\mathcal{A}$ *has queried on.*
4. ***Output Phase****: Finally, the algorithm outputs a circuit* f^**.*

The adversary wins the game if and only if

$$\mathsf{Extract}(\mathsf{xk}, f^*) \notin Q \ \wedge \ \Pr_{x \leftarrow D_f}[f^*(x) = f(x)] \geq \gamma$$

We say a watermarking scheme has γ-unremovability with respect to $\mathcal{F}, \mathcal{D}$, if for all QPT $\mathcal{A}$, it wins the above game with negligible probability in λ.

Remark 3. Here, we only consider a weaker security notion where a quantum adversary only has classical oracle access in the query phase. We claim it is practical and good enough in most of the settings since the watermarking key mk is only in the hands of the challenger: whenever adversary queries $\mathsf{Mark}(\mathsf{mk}, f, \cdot)$, the challenger can always measure this query.

Remark 4. Watermarking primitives should also satisfy 'meaningfulness' property [GKM+19] but since we do not use this property in our construction, we omit it here.

4 Approximating Threshold Implementation

By applying $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ and checking if the outcome is greater than or smaller than γ, we get a approximated threshold implementation $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$. Here, we use $(\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}, \mathbf{I} - \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma})$ to denote this binary POVM.

Theorem 3 gives the following theorem on approximating threshold implementation:

Theorem 4. *For any $\epsilon, \delta, \gamma, \mathcal{P}, D$, the algorithm of measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ that satisfies the followings:*

- *For all quantum state ρ, $\mathrm{Tr}[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} \cdot \rho] \geq \mathrm{Tr}[\mathsf{TI}_{\gamma}(\mathcal{P}_D) \cdot \rho] - \delta$.*
- *By symmetry, for all quantum state ρ, $\mathrm{Tr}[\mathsf{TI}_{\gamma-\epsilon}(\mathcal{P}_D) \cdot \rho] \geq \mathrm{Tr}[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma} \cdot \rho] - \delta$.*
- *For all quantum state ρ, let ρ' be the collapsed state after applying $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ on ρ (conditioned on outcome 1). Then, $\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(P_D) \cdot \rho'] \geq 1 - 2\delta$.*
- *The expected running time is the same as $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$.*

Intuitively the theorem says that if a quantum state ρ has weight p on eigenvectors with eigenvalues at least γ, the measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon}$ with probability at least $p - \delta$ will produce a collapsed state which has weight $1 - 2\delta$ on eigenvectors with eigenvalues at least $\gamma - 2\epsilon$. Also note that the running time is proportional to $\mathsf{poly}(1/\epsilon, 1/(\log \delta))$, which is a polynomial in λ as long as ϵ is any inverse polynomial and δ is any inverse sub-exponential function. The proof of the above theorem is in the full version.

We can also consider approximating the measurements on a bipartite (possibly entangled) quantum state. In this case, we will prove a similar statement as Theorem 4.

Lemma 3. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two collections of projective measurements and D_1 and D_2 be any probability distributions defined on the index set of $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. For any $0 < \epsilon, \delta, \gamma < 1$, the algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ and $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ satisfy the followings:*

- *For any bipartite (possibly entangled, mixed) quantum state $\rho \in \mathscr{H}_{\mathcal{L}} \otimes \mathscr{H}_{\mathcal{R}}$,*

$$\mathrm{Tr}\left[\left(\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}\right)\rho\right] \geq \mathrm{Tr}\left[\left(\mathsf{TI}_{\gamma}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma}(\mathcal{P}_{D_2})\right)\rho\right] - 2\delta.$$

- *For any (possibly entangled, mixed) quantum state ρ, let ρ' be the collapsed state after applying $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on ρ (and normalized). Then,*

$$\mathrm{Tr}\left[\left(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})\right)\rho'\right] \geq 1 - 4\delta.$$

We defer the proof of the above Lemma to the full version.

5 Quantum Copy-Protection Scheme

Let λ be the security parameter. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a class of circuits. We assume $\mathcal{F}$ is quantum unlearnable with respect to $\mathcal{D}$ (see Definition 15) and can be computed by polynomial-sized classical circuits. The construction for quantum copy-protection of function class $\mathcal{F}_\lambda$ is defined in Fig. 1.

Note that this construction works for general quantum unlearnable function families as well. By simply changing the notation in the proof to that in the general quantum unlearnability case, we prove it for general quantum unlearnable function families. More discussion will be given in the full version.

Oracle Heuristics In practice we use a quantum-secure PRF [Zha12] to implement function g; and we use quantum-secure (classical) VBB obfuscation to implement each of $(\mathcal{O}_1, \mathcal{O}_2, U_A, U_{A^\perp})$. We can replace VBB obfuscation programs with oracles that only allow black-box access by the security of VBB obfuscation; afterwards, we can also replace PRF g with a real random function by the property of PRF. The heuristic analysis is straightforward and we omit them here.

5.1 Correctness and Efficiency

Correctness For the quantum program $\left(\rho = |A\rangle\langle A|, \{U_x\}_{x\in[N]}\right)$ produced by the $\mathsf{Generate}$ algorithm, it performs the following computation:

1. Make an oracle query $\mathcal{O}_1$ on the state $|0\rangle|x\rangle|A\rangle$, the resulting state is statistically close to $|y_1\rangle|x\rangle|A\rangle$. Note that $|A\rangle$ with overwhelming probability $1 - 1/|A|$ contains a non-zero vector in A. It measures y_1, which is $y_1 = f(x) \oplus g(x)$.
2. It then prepares a state by applying QFT on the third register and the resulting state is is statistically close to $|0\rangle|x\rangle\left|A^\perp\right\rangle$. It makes an oracle query $\mathcal{O}_2$ on the state $|0\rangle|x\rangle\left|A^\perp\right\rangle$, the resulting state is statistically close to $|y_2\rangle|x\rangle\left|A^\perp\right\rangle$ where $y_2 = g(x)$.

Therefore, with overwhelming probability, the output is $y_1 \oplus y_2 = f(x)$.

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$: The setup algorithm takes in security parameter 1^λ.

- Pick a uniformly random subspace $A \subseteq \mathbb{F}^\lambda$ of dimension $\lambda/2$.
- Output $\mathsf{sk} = A$, where A is described by a set of orthogonal basis vectors.

$\mathsf{Generate}(\mathsf{sk}, f \in \mathcal{F}_\lambda)$: The $\mathsf{Generate}$ algorithm receives $\mathsf{sk} = A$ and a function f from $\mathcal{F}_\lambda$.

- Prepare a subspace state on n qubits corresponding to A, $|A\rangle = \frac{1}{\sqrt{|A|}} \sum_{v \in A} |v\rangle$.
- Generate oracles $U_A, U_{A^\perp}$ which compute subspace membership functions for subspace A and its dual subspace $A^\perp$ respectively.
- Generate oracles $\mathcal{O}_1, \mathcal{O}_2$ such that

$$\mathcal{O}_1(x,v) = \begin{cases} f(x) \oplus g(x) & \text{if } v \in A \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

$$\mathcal{O}_2(x,v) = \begin{cases} g(x) & \text{if } v \in A^\perp \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

  where g is a uniformly random function, with the same input and output length as f.
- Finally, the $\mathsf{Generate}$ algorithm outputs a quantum program $\left(\rho = |A\rangle\langle A|, \{U_x\}_{x\in[N]}\right)$, which describes the following procedure:
  - On input x, prepare the state $|0\rangle\langle 0| \otimes |x\rangle\langle x| \otimes \rho$ and make an oracle query U_A and measure the first register (output register) to get y_1; the remaining state is $|x\rangle\langle x| \otimes \rho'$.
  - Apply QFT on the third register ρ' to get ρ''.
  - Prepare the state $|0\rangle\langle 0| \otimes |x\rangle\langle x| \otimes \rho''$ and make an oracle query $U_{A^\perp}$ and measure the first register to get y_2.
  - Output $y_1 \oplus y_2$.

The description of $\{U_x\}_{x\in[N]}$ requires the oracle of $U_A, U_{A^\perp}$ (or the VBB obfuscations).

Fig. 1. Quantum copy-protection scheme.

Efficiency In $\mathsf{Generate}$ algorithm, as shown in [AC12], given the basis of A, the subspace state $|A\rangle$ can be prepared in polynomial time using QFT. For the oracles $\mathcal{O}_1, \mathcal{O}_2$, it only needs to check the membership of A and $A^\perp$ and compute functions f and g. f can be prepared in polynomial time by definition. As we discussed above, we can prepare the function g as a PRF. Therefore, the oracles $\mathcal{O}_1, \mathcal{O}_2$ can be generated in polynomial time. The $\mathsf{Compute}$ algorithm is clearly efficient.

5.2 Anti-Piracy Security

We show that for a *quantum unlearnable* families of functions $\mathcal{F}$ with respect to $\mathcal{D}$ defined in Definition 15, the quantum copy-protection scheme has anti-piracy security against any quantum polynomial-time adversaries. More formally:

Theorem 5 (Main Theorem). *Let $\mathcal{F}$ be a function families that is γ-quantum-unlearnable with respect to distribution $\mathcal{D}$ (γ is a non-negligible function of λ). The above copy-protection scheme for $\mathcal{F}, \mathcal{D}$ has $(\gamma(\lambda) - 1/\mathsf{poly}(\lambda))$-anti-piracy security, for all polynomial poly.*

In order to describe the quantum query behavior of quantum programs made to oracles, we give the following definitions and notations.

We recall that in Definition 13, a QPT adversary $\mathcal{A}$ in the anti-piracy security game $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, will produce a state σ over registers R_1, R_2 and unitaries $\{U_{R_1,x}\}_{x\in[N]}, \{U_{R_2,x}\}_{x\in[N]}$, the challenger will then perform γ-goodness test on σ using threshold implementations $\mathsf{TI}_\gamma(P_{R_1,f})$ and $\mathsf{TI}_\gamma(P_{R_2,f})$. For simplicity we will describe the unitary ensembles $\{U_{R_1,x}\}_{x\in[N]}$, $\{U_{R_2,x}\}_{x\in[N]}$ as U_{R_1}, U_{R_2} and describe threshold implementations $\mathsf{TI}_\gamma(P_{R_1,f})$, $\mathsf{TI}_\gamma(P_{R_2,f})$ as $\mathsf{TI}_{R_1,\gamma}, \mathsf{TI}_{R_2,\gamma}$. Similarly, let $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ and $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ denote the approximation threshold implementation $\mathsf{ATI}^{\epsilon,\delta}_{R_1,\gamma-\epsilon}$ and $\mathsf{ATI}^{\epsilon,\delta}_{R_2,\gamma-\epsilon}$ respectively, for some inverse polynomial ϵ and inverse subexponential function δ (in other words, $\log(1/\delta)$ is polynomial in λ).

In this particular construction, $\mathcal{A}$'s behavior can be described as follows: $\mathcal{A}$ "splits" the copy-protection state ρ into two potentially entangled states $\sigma[R_1], \sigma[R_2]$. $\mathcal{A}$ prepares $(\sigma[R_1], U_{R_1})$ with oracle access to $(\mathcal{O}_1, \mathcal{O}_2)$ as pirate program P_1; and prepares $(\sigma[R_2], U_{R_2})$ with oracle access $(\mathcal{O}_1, \mathcal{O}_2)$ as pirate program P_2. Therefore, $\mathsf{TI}_{R_b,\gamma}$ and $\mathsf{ATI}_{R_b,\gamma-\epsilon}$ both make oracle queries to $\mathcal{O}_1, \mathcal{O}_2$.

We can assume the joint state of R_1, R_2 has been purified and the overall state is a pure state over register R_1, R_2, R_3 where P_1 has only access to R_1 and P_2 has only access to R_2.

Quantum Query Weight Let σ be any quantum state of R_1, R_2, R_3. We consider the program P_1. P_1 has access to register R_1 and oracle access to $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$. We denote $|\phi_i\rangle$ to be the overall state of registers R_1, R_2, R_3 before P_1 makes i-th query to $\mathcal{O}_1$, *when it applies* $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ *on* $\sigma[R_1]$.

$$|\phi_i\rangle = \sum_{x,v,z} \alpha_{x,v,z} |x, v, z\rangle .$$

where (x, v) is the query to oracle $\mathcal{O}_1$ and z is working space of P_1, the registers of R_2, R_3. Note that when $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ is applied on $\sigma[R_1]$, it in fact applies some unitary and eventually makes a measurement, during which the unitary makes queries to oracles $\mathcal{O}_1, \mathcal{O}_2$. Therefore such a query weight is well-defined.

We denote by $W_{1,A,i}$ to be the sum of squared amplitudes in $|\phi_i\rangle$, which are querying $\mathcal{O}_1$ on input (x, v) such that $v \in A \setminus \{0\}$:

$$W_{1,A,i} = \sum_{x,v,z:v\in A\setminus\{0\}} |\alpha_{x,v,z}|^2$$

Then we sum up all the squared amplitudes $W_{1,A,i}$ in all the queries made by P_1 to $\mathcal{O}_1$, where $v \in A \setminus \{0\}$. We denote this sum as $W_{1,A} = \sum_{i\in[\ell_1]} W_{1,A,i}$, where $\ell_1 = \ell_1(\lambda)$ is the number of queries made by P_1 to $\mathcal{O}_1$.

Similarly, we write $W_{1,A^\perp} = \sum_{i\in[\ell_2]} W_{1,A^\perp,i} = \sum_{i\in[\ell_2]} \sum_{x,v,z:v\in A^\perp\setminus\{0\}} |\alpha_{x,v,z}|^2$ to be the sum of squared amplitudes in $|\phi_i\rangle$ where $v \in A^\perp \setminus \{0\}$, in the ℓ_2 queries made by P_1 to $\mathcal{O}_2$.

Accordingly, for the other program P_2 and threshold implementation $\mathsf{ATI}_{R_2,\gamma-\epsilon}$, we denote these sums of squared amplitudes as $W_{2,A} = \sum_{i\in[m_1]} W_{2,A,i}$ and $W_{2,A^\perp} = \sum_{i\in[m_2]} W_{2,A^\perp,i}$, where m_1, m_2 are the number of queries made by P_2 to oracles $\mathcal{O}_1, \mathcal{O}_2$ respectively.

Case One. Fixing a function f, let $(\sigma, U_{R_1}, U_{R_2})$ be the two programs output by the adversary which are both tested γ-good with respect to f, D_f with some non-negligible probability.

Let $\mathcal{O}_\perp$ be an oracle that always outputs $\perp$. We hope one of the following events will happen:

1. The program $(\sigma[R_1], U_{R_1})$ with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ is tested $(\gamma-2\epsilon)$-good with respect to f, D_f, with non-negligible probability.
2. The program $(\sigma[R_2], U_{R_2})$ with oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$ is tested $(\gamma-2\epsilon)$-good with respect to f, D_f, with non-negligible probability.

Let $\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon}$ be the same as $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ except with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ and $\widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}$ be the same as $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ except with oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$. Similarly, let $\widetilde{\mathsf{TI}}_{R_b,\gamma-2\epsilon}$ be the same threshold implementation as $\mathsf{TI}_{R_b,\gamma-2\epsilon}$ except with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ and $\mathcal{O}_\perp, \mathcal{O}_2$ respectively.

Since $(\sigma[R_1], U_{R_1})$ and $(\sigma[R_2], U_{R_2})$ are both γ-good with respect to $f, \mathcal{D}_f$ with non-negligible probability, for some non-negligible function $\beta(\cdot)$,

$$\mathrm{Tr}[(\mathsf{TI}_{R_1,\gamma} \otimes \mathsf{TI}_{R_2,\gamma}) \cdot \sigma] \geq \beta(\lambda)$$

From the property of the approximated threshold implementation (Lemma 3),

$$\mathrm{Tr}[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \mathsf{ATI}_{R_2,\gamma-\epsilon}) \cdot \sigma] \geq \beta(\lambda) - 2\delta$$

Thus, for any $b \in \{1, 2\}$, we have $\mathrm{Tr}[\mathsf{ATI}_{R_b,\gamma-\epsilon} \cdot \sigma[R_b]] \geq \beta(\lambda) - 2\delta$. Since δ is negligible, both probabilities are still non-negligible.

We then define the following two events:

E_1 : Let E_1 be the event denotes $\mathrm{Tr}[\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \cdot \sigma[R_1]]$ is non-negligible. If E_1 happens, by Theorem 4,

$$\mathrm{Tr}\left[\widetilde{\mathsf{TI}}_{R_1,\gamma-2\epsilon} \cdot \sigma[R_1]\right] \geq \mathrm{Tr}\left[\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \cdot \sigma[R_1]\right] - \delta$$

which is still non-negligible. In other words, $(\sigma[R_1], U_{R_1})$ with oracle access to $\mathcal{O}_1, \mathcal{O}_\perp$ is tested $(\gamma - 2\epsilon)$-good with respect to f, D_f with non-negligible probability.

E_2 : Similarly, define E_2 as the program $(\sigma[R_2], U_{R_2})$ with oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$ is $(\gamma - 2\epsilon)$-good with respect to f, D_f with non-negligible probability.

Case Two. Fixing a function f, let $(\sigma, U_{R_1}, U_{R_2})$ be the two programs output by the adversary which are both γ-good with respect to f, D_f, with non-negligible probability.

If $\mathsf{E}_1 \vee \mathsf{E}_2$ does not happen, we are in the case $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$. By definition, there exist negligible functions $\mathsf{negl}_1, \mathsf{negl}_2$ such that

$$\mathrm{Tr}\left[\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \cdot \sigma[R_1]\right] \leq \mathsf{negl}_1(\lambda), \qquad \mathrm{Tr}\left[\widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon} \cdot \sigma[R_2]\right] \leq \mathsf{negl}_2(\lambda).$$

We look at the following thought experiments:

1. We apply $\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \mathsf{ATI}_{R_2,\gamma-\epsilon}$ on σ, by Lemma 3, there exists a non-negligible function $\beta(\cdot)$ such that

$$\mathrm{Tr}\left[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \mathsf{ATI}_{R_2,\gamma-\epsilon}) \cdot \sigma\right] \geq \beta(\lambda) - 2\delta.$$

2. We apply $\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}$ on σ. We have,

$$\mathrm{Tr}\left[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes \widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}) \cdot \sigma\right] \leq \mathrm{Tr}\left[(I \otimes \widetilde{\mathsf{ATI}}_{R_2,\gamma-\epsilon}) \cdot \sigma\right] \leq \mathsf{negl}_2(\lambda).$$

3. Note that in 1 and 2, the only difference is the oracle access: in 1, it has oracle access to $\mathcal{O}_1, \mathcal{O}_2$; in 2, it has oracle access to $\mathcal{O}_\perp, \mathcal{O}_2$. Let σ' be the state which we apply $(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes I)$ on σ and obtain a outcome 1, which happens with non-negligible probability. Let $W_{2,A}$ be the query weight defined on the state σ'. We know that $W_{2,A}$ can not be negligible otherwise by Theorem 1 (BBBV), the probability difference in 1 and 2 can not be non-neglibile.
   Define M_{R_2} be the operator that measures a random query of $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ to $\mathcal{O}_1$ and the query (x, v) satisfies $v \in A \setminus \{0\}$. By the above discussion, there exists a non-negligible function $\beta_1(\cdot)$,

$$\mathrm{Tr}\left[(\mathsf{ATI}_{R_1,\gamma-\epsilon} \otimes M_{R_2}) \cdot \sigma\right] \geq \beta_1(\lambda).$$

4. We apply $\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \otimes M_{R_2}$ on σ. We have,

$$\mathrm{Tr}\left[(\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \otimes M_{R_2}) \cdot \sigma\right] \leq \mathrm{Tr}\left[(\widetilde{\mathsf{ATI}}_{R_1,\gamma-\epsilon} \otimes I) \cdot \sigma\right] \leq \mathsf{negl}_1(\lambda).$$

5. By a similar argument of 3, let M_{R_1} be the operator that measures a random query of $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ to $\mathcal{O}_2$ and the query (x,v) satisfies $v \in A^\perp \setminus \{0\}$. There exists a non-negligible function $\beta_2(\cdot)$,

$$\mathrm{Tr}\left[(M_{R_1} \otimes M_{R_2}) \cdot \sigma\right] \geq \beta_2(\lambda).$$

Thus, in the case, one can extract a pair of vectors $(u,v) \in (A \setminus \{0\}) \times (A^\perp \setminus \{0\})$ with non-negligible probability. To conclude it, we have the following lemma,

Lemma 4. *Fixing a function f, let $(\sigma, U_{R_1}, U_{R_2})$ be the two programs output by the adversary which are both γ-good with respect to f, D_f, with non-negligible probability. If $\mathsf{E}_1 \vee \mathsf{E}_2$ does not happen, by randomly picking and measuring a query of $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ to $\mathcal{O}_2$ and a query of $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ to $\mathcal{O}_1$, one can obtain a pair of vectors $(u,v) \in (A \setminus \{0\}) \times (A^\perp \setminus \{0\})$ with non-negligible probability.*

Then we show a reduction to violate unlearnability in case of E_1 or E_2 and a reduction to violate direct product hardness in case of $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$. We have the following lemmas:

Lemma 5. *Let $\Pr[\mathsf{E}_1]$ be the probability of E_1 taken over all randomness of $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. If $\Pr[\mathsf{E}_1]$ is non-negligible, there exists an adversary $\mathcal{A}_1$ that wins $\mathsf{LearningGame}^{\mathcal{A}_1}_{\mathcal{F},\mathcal{D},\gamma-2\epsilon}(1^\lambda)$ with non-negligible probability.*

Proof. The challenger in the copy-protection security game plays as the quantum unlearnability adversary $\mathcal{A}_1$ for function $f \leftarrow \mathcal{F}$, given only black-box access to f; we denote this black box as oracle $\mathcal{O}_f$, which on query $|x,z\rangle$, answers the query with $|x, f(x)+z\rangle$.

Next, we show that $\mathcal{A}_1$ can simulate the copy-protection security game for $\mathcal{A}$ using the information given and uses $\mathcal{A}$ to quantumly learn f. $\mathcal{A}_1$ samples random $\lambda/2$-dimensional subspace A over $\mathbb{F}$ and prepares the membership oracles (two unitaries) $U_A, U_A^\perp$ as well as state $|A\rangle$.

Using $U_A, U_A^\perp$ and given oracle access to f in the unlearnability game, $\mathcal{A}_1$ simulates the copy-protection oracles $\mathcal{O}_1, \mathcal{O}_2$ for $\mathcal{A}$ in the query phase of anti-piracy game.

There is one subtlety in the proof: $\mathcal{A}_1$ needs to simulate the oracles in the anti-piracy game slightly differently: $\mathcal{A}_1$ simulates the oracles with their functionalities partially swapped:

$$\mathcal{O}'_1(x,v) = \begin{cases} g(x) & \text{if } v \in A \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

$$\mathcal{O}'_2(x,v) = \begin{cases} f(x) \oplus g(x) & \text{if } v \in A^\perp \text{ and } v \neq 0, \\ \perp & \text{otherwise.} \end{cases}$$

That is, a random function $g(x)$ is output when queried on $u \in A \setminus \{0\}$, and $f(x) \oplus g(x)$ is output when queried on $u \in A^\perp \setminus \{0\}$. The distributions of $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{O}'_1, \mathcal{O}'_2$ are identical. Note that $g(x)$ can be simulated by a quantum secure

PRF or a $2t$-wise independent hash function where t is the number of oracle queries made by $\mathcal{A}$ [Zha12].

In the output phase, $\mathcal{A}$ outputs $(\sigma, U_{R_1}, U_{R_2})$ and sends to $\mathcal{A}_1$. $\mathcal{A}_1$ simply outputs $(\sigma[R_1], U_{R_1})$ with oracle access to $\mathcal{O}'_1, \mathcal{O}_\perp$. The program does not need access to oracle f because $\mathcal{O}'_1$ is only about $g(\cdot)$ and $\mathcal{O}_\perp$ is a dummy oracle. If E_1 happens, the program is a $(\gamma - 2\epsilon)$-good with non-negligible probability, by the definition of E_1. Because $\Pr[E_1]$ is also non-negligible, $\mathcal{A}_1$ breaks $(\gamma - 2\epsilon)$-quantum-unlearnability of $\mathcal{F}, \mathcal{D}$. □

Lemma 6. *Let* $\Pr[\mathsf{E}_2]$ *be the probability of* E_2 *taken over all randomness of* $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. *If* $\Pr[\mathsf{E}_2]$ *is non-negligible, there exists an adversary* $\mathcal{A}_2$ *that wins* $\mathsf{LearningGame}^{\mathcal{A}_2}_{\mathcal{F},\mathcal{D},\gamma-2\epsilon}(1^\lambda)$ *with non-negligible probability.*

Proof (Proof Sketch). The proof is almost identical to the proof for Lemma 6 except oracles $\mathcal{O}_1, \mathcal{O}_2$ are simulated in the same way as that in the construction. $\mathcal{O}_1(x, v)$ outputs $f(x) \oplus g(x)$ if $v \in A \setminus \{0\}$, and otherwise outputs $\perp$. Similarly, $\mathcal{O}_2(x, v)$ outputs $g(x)$ if $v \in A^\perp \setminus \{0\}$, and otherwise outputs $\perp$ □

As discussed above, if $\Pr[\mathsf{E}_1 \vee \mathsf{E}_2]$ is non-negligible, we can break the quantum unlearnability. Otherwise, $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ is overwhelming. We show that in the case, one can use the adversary $\mathcal{A}$ to break the direct-product problem Theorem 2.

Lemma 7. *Let* $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ *be the probability taken over all randomness of the game* $\mathsf{CopyProtectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. *If* $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ *is non-negligible, there exists an adversary* $\mathcal{A}_3$ *that breaks the direct-product problem.*

Proof. The challenger in the copy-protection security game plays as the adversary in breaking direct-product problem, denoted as $\mathcal{A}_3$. In the reduction, $\mathcal{A}_3$ is given the access to membership oracles $U_A, U_A^\perp$ and one copy of $|A\rangle$.

Next, we show that $\mathcal{A}_3$ can simulate the anti-piracy security game for $\mathcal{A}$ using the information given and uses $\mathcal{A}$ to obtain the two vectors. $\mathcal{A}_3$ samples $f \leftarrow \mathcal{F}$, and simulates a γ-anti-piracy game, specifically simulating the copy-protection oracle $\mathcal{O}_1, \mathcal{O}_2$ for adversary $\mathcal{A}$. In the output phase, $\mathcal{A}$ outputs $(\sigma, U_{R_1}, U_{R_2})$.

$\mathcal{A}_1$ upon taking the output, it randomly picks and measures a query of $\mathsf{ATI}_{R_1,\gamma-\epsilon}$ to $\mathcal{O}_2$ and a query of $\mathsf{ATI}_{R_2,\gamma-\epsilon}$ to $\mathcal{O}_1$, and obtain a pair of vectors (u, v). If $\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2$ happens. By Lemma 4, (u, v) breaks the direct-product problem with non-negligible probability. Since $\Pr[\bar{\mathsf{E}}_1 \wedge \bar{\mathsf{E}}_2]$ is non-negligible, the overall probability is non-negligible. □

Note that the proof does not naturally extend to q-collusion resistant anti-piracy. We leave this as an interesting open problem.

6 Quantum Copy-Detection

6.1 Construction

We construct a copy-detection scheme for a watermarkable function family $\mathcal{F}$ with respect to an input distribution $\mathcal{D}$. Let QM and WM be a public key quantum money scheme and a publicly extractable watermarking scheme for $\mathcal{F}, \mathcal{D}$,

whose serial number space $\mathcal{S}_\lambda$ of QM is a subset of the message space $\mathcal{M}_\lambda$ of WM. We construct a copy-detection scheme in Fig. 2. The definition of quantum money schemes, our general scheme and full proofs are in the full version of this paper.

$\mathsf{Setup}(1^\lambda)$: it runs $\mathsf{WM.Setup}(1^\lambda)$ to get xk, mk, let $\mathsf{sk} = \mathsf{mk}$ and $\mathsf{pk} = \mathsf{xk}$.
$\mathsf{Generate}(\mathsf{sk}, f)$:
- it runs $\mathsf{QM.Gen}(1^\lambda)$ to get a money state $|\$\rangle$ and a serial number s (by applying $\mathsf{QM.Ver}$ to the banknote);
- let $\widetilde{f} = \mathsf{WM.Mark}(\mathsf{mk}, f, s)$ which is classical;
- it outputs the quantum state $\rho_f = (\widetilde{f}, |\$\rangle)$, and $\{U_{f,x}\}_{x\in[N]}$;
- let $\{U_{f,x}\}_{x\in[N]}$ describe the following unitary: on input a quantum state ρ, treat the first register as a classical function g, compute $g(x)$ in superposition.

$\mathsf{Check}(\mathsf{pk}, (\rho_f, \{U_{f,x}\}_{x\in[N]}))$:
- it parses and measures the first register, which is $(f', |\$'\rangle)$;
- it checks if $\mathsf{QM.Ver}(|\$'\rangle)$ is valid and it gets the serial number s';
- it then checks if $s' = \mathsf{WM.Extract}(\mathsf{pk} = \mathsf{xk}, f')$;
- if all the checks pass, it outputs 1; otherwise, it outputs 0.

Fig. 2. Quantum copy-detection scheme.

6.2 Efficiency and Correctness

First, for all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \in \mathcal{F}_\lambda$, the copy-detection program is $(\rho_f, \{U_{f,x}\}_{x\in[N]})$. We have $\mathsf{Compute}(\rho_f, \{U_{f,x}\}_{x\in[N]}, x) = \tilde{f}(x)$, where $\tilde{f} = \mathsf{WM.Mark}(\mathsf{mk}, f, s)$ for some serial number s. From the correctness of WM, it satisfies the correctness of copy-detection.

The correctness of Check comes from the correctness of $\mathsf{WM.Extract}$ and **unique serial number** property of QM. Check is a projection since $\mathsf{QM.Ver}$ is also a projection. Efficiency is straightforward.

6.3 Security

Theorem 6. *Assume* QM *is a quantum money scheme and* WM *is a watermarking scheme for* $\mathcal{F}, \mathcal{D}$ *with* γ*-unremovability, the above copy-detection scheme for* $\mathcal{F}, \mathcal{D}$ *has* γ*-copy-detection-security.*

Proof. Let $\mathcal{A}$ be a QPT algorithm that tries to break the security of the copy-detection scheme. Let $(\sigma, U_{R_1}, U_{R_2})$ be the programs output by $\mathcal{A}$ which wins the game $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}$. To win the game, the program $(\sigma, U_{R_1}, U_{R_2})$ should pass the following two tests:

1. Apply the projective measurement (defined by $\mathsf{Check}(\mathsf{pk}, \cdot)$) on both $\sigma[R_1]$ and $\sigma[R_2]$, and both outcomes are 1.
2. Let σ' be the state that passes step 1. Then both programs $(\sigma'[R_1], U_{R_1})$, $(\sigma'[R_2], U_{R_2})$ are tested to be γ-good with non-negligible probability.

In our construction, Check first measures the program registers. The resulting state is $\tilde{f}_1, \tilde{f}_2, \sigma$, where $\tilde{f}_1, \tilde{f}_2$ are supposed to be classical (marked) circuits that computes f and σ are (possibly entangled) states that are supposed to be quantum money state for each of the program.

Next, Check applies $\mathsf{QM.Ver}$ on both registers of σ and computes serial numbers. Define S_b be the random variable of $\mathsf{QM.Ver}$ applying on $\sigma[R_b]$ representing the serial number of ρ_b, for $b = 1, 2$. Define S be the random variable of $\mathsf{QM.Ver}(|\$\rangle)$ representing the serial number of the quantum money state in the $\mathsf{Generate}$ procedure.

Define E be the event that both $\mathsf{WM.Extract}(\mathsf{xk}, \tilde{f}_b) = S_b$ and at least one of S_1, S_2 is not equal to S. Define E' be the event that both S_1, S_2 are equal to S and both $\mathsf{WM.Extract}(\mathsf{xk}, \tilde{f}_b) = S_b$. If $\tilde{f}_1, \tilde{f}_2, \sigma$ passes the step 1, exactly one of E and E' happens.

In step 2, it simply tests if $\tilde{f}_1$ and $\tilde{f}_2$ are γ-good with respect to f, D_f. Since $\tilde{f}_1, \tilde{f}_2$ are classical circuits, it is equivalent to check whether they work correctly on at least γ fraction of all inputs. If it passes step 2, we have for all $b \in \{1, 2\}$, $\Pr_{x \leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma$.

Therefore, the probability of $\mathcal{A}$ breaks the security game is indeed,

$$\begin{aligned}
&\Pr_{(\tilde{f}_1, \tilde{f}_2, \sigma)}\left[(E \vee E') \wedge \forall b, \Pr_{x \leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] \\
\leq &\Pr_{(\tilde{f}_1, \tilde{f}_2, \sigma)}\left[E \wedge \forall b, \Pr_{x \leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] + \Pr_{(\tilde{f}_1, \tilde{f}_2, \sigma)}[E']
\end{aligned}$$

Note that the probability is taken over the randomness of $\mathsf{CopyDetectionGame}^{\mathcal{A}}_{\mathcal{F}, \mathcal{D}, \gamma}$. Next we are going to show both probabilities are negligible, otherwise we can break the quantum money scheme or watermarking scheme.

Claim 1. $\Pr_{(\tilde{f}_1, \tilde{f}_2, \sigma)}[E'] \leq \mathit{negl}(\lambda)$.

Proof. It corresponds to the security game of the quantum money scheme. Assume $\Pr[E']$ is non-negligible, we can construct an adversary $\mathcal{B}$ for the quantum money scheme with non-negligible advantage. Given a quantum money state $|\$\rangle$, the algorithm $\mathcal{B}$ simulates the challenger for the copy-detection game and can successfully 'copy' a money state. □

Claim 2. $\Pr_{(\tilde{f}_1, \tilde{f}_2, \sigma)}\left[E \wedge \forall b, \Pr_{x \leftarrow D_\lambda}[\tilde{f}_b(x) = f(x)] \geq \gamma\right] \leq \mathit{negl}(\lambda)$.

Proof. It corresponds to the security game of the underlying watermarking scheme. Since if E happens, at least one of the circuit has different mark than s and it satisfies the correctness requirement. □

Thus, the probability of $\mathcal{A}$ breaks the game is negligible. □

Acknowledgements

We thank Paul Christiano for suggesting the idea of quantum copy-protection based on [AC12] hidden subspace oracles.

J. L., Q. L., M. Z. and R. Z.'s research is supported by NSF Grant; S. A. is supported by Vannevar Bush Faculty Fellowship from the US Department of Defense, the Simons Foundation's It from Qubit Collaboration, and a Simons Investigator Award.

References

Aar04. Scott Aaronson. Limitations of quantum advice and one-way communication. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 320–332. IEEE, 2004.

Aar09. Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242. IEEE, 2009.

AC12. Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 41–60. ACM, 2012.

AGKZ20. Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 255–268. Association for Computing Machinery, 2020.

AP20. Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing, 2020.

BBBV97. Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, Oct 1997.

BDGM20. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. https://eprint.iacr.org/2020/1024.

BDS16. Shalev Ben-David and Or Sattath. Quantum tokens for digital signatures. *arXiv preprint arXiv:1609.09047*, 2016.

BGI+01. Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual International Cryptology Conference*, pages 1–18. Springer, 2001.

BGMZ18. James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Preventing zeroizing attacks on ggh15. In *Proceedings of TCC 2018*, 2018.

BJL+21. Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions, 2021.

BL19. Anne Broadbent and Sébastien Lord. Uncloneable quantum encryption via random oracles. *IACR Cryptology ePrint Archive*, 2019:257, 2019.

BP15. Nir Bitansky and Omer Paneth. On non-black-box simulation and the impossibility of approximate obfuscation. *SIAM Journal on Computing*, 44(5):1325–1383, 2015.

CHN+18. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM Journal on Computing*, 47(6):2157–2202, 2018.

CMP20. Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model, 2020.

GKM+19. Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J Wu. Watermarking public-key cryptographic primitives. In *Annual International Cryptology Conference*, pages 367–398. Springer, 2019.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–621. IEEE, 2017.

GZ20. Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020. https://eprint.iacr.org/2020/877.

KNY20. Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions, 2020.

KW17. Sam Kim and David J Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *Annual International Cryptology Conference*, pages 503–536. Springer, 2017.

KW19. Sam Kim and David J Wu. Watermarking prfs from lattices: Stronger security via extractable prfs. In *Annual International Cryptology Conference*, pages 335–366. Springer, 2019.

QWZ18. Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking prfs under standard assumptions: Public marking and security with extraction queries. In *Theory of Cryptography Conference*, pages 669–698. Springer, 2018.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611. IEEE, 2017.

Zha12. Mark Zhandry. How to construct quantum random functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 679–687. IEEE, 2012.

Zha20. Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. Cryptology ePrint Archive, Report 2020/1191, 2020. https://eprint.iacr.org/2020/1191.