New Approaches for Quantum Copy-Protection

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

[1] The University of Texas at Austin
{aaronson, jiahui, rzzhang}@cs.utexas.edu
[2] Princeton University & NTT Research, USA
mzhandry@princeton.edu
[3] Princeton University, USA
qipengl@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 how to achieve 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 improves on Aaronson [CCC'09], 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 general, applicable to many recent watermarking schemes.

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, any *learnable* program—that is, a program whose description can

be learned from just it's input/output behavior—cannot be copy-protected. Indeed, an attacker, given the (copy-protected) code for the program can just query the code on several inputs, and learn the original program from the results. The original program can then be copied indefinitely. A more recent result of Ananth and La Placa [AP20] shows, under certain computational assumptions, that even 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 the general. Worse, even for programs that are not subject to the impossibility result, it remains unclear how to even heuristically 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 guarantees that such copying can be detected. Also, their construction only works for a certain class of "evasive" functions, which only 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 very 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 for much 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, but 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 evasive functions, by copy-detecting arbitrary watermarkable functions. For our purposes, watermarkable functions are those that can have a publicly observable "mark" embedded into the program, such that it is infeasible

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

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 look at one attempt of defining quantum copy-protection. We say an adversary successfully pirates a quantum program for computing function f, if it outputs two quantum programs σ_1, σ_2, each of them able to compute f correctly with probability greater than some threshold. Consider the following case. Let f be a signing algorithm with a particular signing key hard-coded. Suppose that there are many valid signatures for each message. Consider a hypothetical adversary which "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 many applications, but this adversary would not be ruled out by the usual security notions.

Another example is 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: namely they can only decrypt correctly on ciphertexts of m_0, m_1; for ciphertexts of other messages, they output junk. 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 some applications.

Similar definitonal issues were discussed in [GKM+19], but in the context of watermarking primitives. As we will see, watermarking is closely related to copy-detection and copy-protection.

Our solution is to define "compute f correctly" by a 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 verification 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 $\mathsf{Ver}(\mathsf{vk}, m, \cdot\,; r)$.

Unfortunately, formalizing these other definitions can still be tricky. For example, we want that the adversary can not take a program for f and produce

two programs that each computes f correctly on half inputs of the domain. In this setting, we would naturally say that a program is good if it correctly computes the function f with probability $1/2$. However the definition becomes problematic. 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. Now, the two halves of this bipartite system each has probability $1/2$ of outputting the right answer on a random input. Thus, both halves would naturally be considered to compute correctly, according to this definition. Therefore, any security definition like this is trivially false.

For another example, consider the adversary produces $\frac{1}{\sqrt{3}}|P\rangle|P\rangle + \frac{\sqrt{2}}{\sqrt{3}}|D\rangle|D\rangle$. The two halves of this bipartite system each has probability $1/3$ of outputting the right answer on a random input. However, both halves can successfully answer all inputs correctly at the same time, with probability $1/3$. Thus, it is secure under the security definition above, but the adversary actually perfectly pirates the program with some constant probability.

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 a eigenvector with eigenvalue p, the 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. Using similar ideas, we define quantum unlearnability of programs, and quantum copy-detection.

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

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

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 which leads 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 any 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 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 with token $|\mathsf{sig}\rangle$ one can only produce one of the classical signatures, a user can still query both oracles $\mathcal{O}_1, \mathcal{O}_2$ multiple times. To obtain $H(x)$, a user can simply 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 $(\sigma_1, \mathcal{O}_1, \mathcal{O}_\bot)$ would have almost identical output distribution. Moreover, $(\sigma_1, \mathcal{O}_1, \mathcal{O}_\bot)$ 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 $(\sigma_2, \mathcal{O}_\bot, \mathcal{O}_2)$ can be used to break the unlearnability of f.

Since f is unlearnable, the above two cases can not happen. We show under this case, we can extract signatures of 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. This 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.

Our Copy-Detection Scheme. We construct a copy-detection scheme for any function family that can be watermarked. A watermarking scheme roughly consists 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 outputs 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 $\hat{f}$ with a different 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: Gen and Ver. Gen takes a security parameter and outputs a quantum banknote $|\$\rangle$. Ver is public, takes a quantum money banknote, and outputs either a serial number of that banknote or $\perp$ indicating it is an invalid banknote. The security requires no efficient adversary could use $|\$\rangle$ to prepare $|\$_1\rangle |\$_2\rangle$ 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 the copy-detected 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 than s.

We show 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 cannot 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 can 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] that 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] introduced unclonable encryption. They construct schemes for encoding classical plaintexts into quantum ciphertexts, which prevents copying of encrypted data. Unclonable encryption can be seen as copy-protecting a unit of functional information simpler than a function. [GZ20] introduced another new notion, unclonable decryption keys; in contrast to making the ciphertext unclonable as in [BL19], they construct schemes where the decryption key is unclonable, therefore allowing only one decryptor to decrypt successfully at a time. A more recent work is [CMP20], giving 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.

Quantum Money Quantum money was first proposed by Wiesner in around 1970; [Wie83] gave a first private-key quantum money scheme based on conjugate coding. Aaronson [Aar09] gave a first public-key quantum money scheme; he proved that it is possible to construct the secure public-key quantum money relative to a quantum oracle. However, his explicit scheme was broken by Lutomirski et al. [LAF+09]. Later, Aaronson and Christiano [AC12] proposed a secure public-key quantum money scheme relative to a classical oracle. Zhandry [Zha19] investigated a kind of collision-free quantum money called quantum lightning and the win-win relationship between the security of signatures/hash functions and quantum money; [Zha19] also instantiated the quantum money scheme of [AC12] with quantum-secure indistinguishability obfuscation. Kane [Kan18] showed a new approach for public-key quantum money using modular forms. Ji et al. [JLS18] defined the pseudorandom quantum state (PRS) and gave a private-key quantum money scheme based on PRS. Recently, Peter Shor [Sho20] proposed a public-key quantum money scheme based on the hardness of a lattice problem.

Another interesting circumstance to consider is classically verifiable quantum money introduced in [Gav12]. [RS19] gave a construction for semi-quantum money which can be verified with a protocol over classical channels.

One-time Programs and One-time Memory Another idea of copy-protecting softwares is through one-time program, introduced in [GKR08]. One-time programs can be executed on only one single input and nothing other than the result of this computation is leaked. Quantum one-time programs are discussed in [BGS13], showing that any quantum circuit can be compiled into a one-time program assuming only the same basic one-time memory devices used for classical circuits. [LSZ20] constructs one-time programs from quantum-accessible one-time memories where the view of an adversary, despite making quantum queries, can be simulated by making only classical queries to the ideal functionality.

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 other cryptographic functionalities such as encryption and signature.

2 Preliminaries

We use λ as 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 use $\mathsf{negl}(x)$ to denote a negligible function. We use QPT to denote quantum polynomial time.

2.1 Quantum Computation

We give some basic definitions of quantum computation and quantum information in Appendix A. 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 totally 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 $\|\tilde{\rho}-\rho\|_{\mathrm{tr}} \leq \sqrt{\epsilon}$.*

2.2 Quantum Oracle Algorithm

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

Definition 1 (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, the word 'oracle' means a 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). The query complexity of a quantum oracle algorithm is the number of $\mathcal{O}$ access.

In the analysis of security of the copy-protection scheme in Section 5.2, we will use the theorem from [BBBV97] to bound the change in adversary's state when we change the oracle's input-output at 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 problem, which are key components of quantum signature token scheme by Ben-David and Sattath [BDS16] and also our quantum copy-protection scheme.

Definition 2 (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 3 (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 4 (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 $v \in A^\perp$.

The hardness of the direct-product problem was proved by Ben-David and Sattath [BDS16], used for 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". Our copy-protection scheme works for general signature token schemes. To keep the statement and proof 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 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 5 (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.*

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

2.4 Measurement Implementation

The following definitions and lemmas are introduced by Zhandry [Zha20].

Definition 6 (Controlled Projection). *Let* $\mathcal{P} = \{\mathcal{P}_i\}_{i\in\mathcal{I}}$ *be a collection of projective measurement over a Hilbert space* $\mathcal{H}$, *where* $\mathcal{P}_i = (P_i, Q_i)$ *for* $i \in \mathcal{I}$. *Let* D *be a distribution with a random coin set* $\mathcal{R}$. *We define the controlled projection, denoted* $\mathsf{CProj}_{\mathcal{P},D} = (\mathsf{CProj}^0_{\mathcal{P},D}, \mathsf{CProj}^1_{\mathcal{P},D})$ *as the follows:*

$$\mathsf{CProj}^0_{\mathcal{P},D} := \sum_{r\in\mathcal{R}} |r\rangle\langle r| \otimes P_{D(r)} \qquad \mathsf{CProj}^1_{\mathcal{P},D} := \sum_{r\in\mathcal{R}} |r\rangle\langle r| \otimes Q_{D(r)}$$

In other words, $\mathsf{CProj}_{\mathcal{P},D}$ uses the random coins r as a control and decides which projective measurement to be applied on the system. That is, $\mathsf{CProj}_{\mathcal{P},D}$ implements the following mixed projective measurement, which is a POVM $\mathcal{P}_D = (P_D, Q_D)$ where $P_D = \sum_{i\in\mathcal{I}} \Pr[i \leftarrow D]\, P_i$ and $Q_D = \sum_{i\in\mathcal{I}} \Pr[i \leftarrow D]\, Q_i$.

For example, D generates a random message m and a random encryption c of this message m. In this case, $\mathcal{I} = \{(m, c)\}$ for all messages and ciphertexts. $\mathcal{P}_{(m,r)} = (P_{(m,r)}, Q_{(m,r)})$ simply means trying to decrypt a ciphertext c and check if the resulting message is equal to m.

Definition 7 (Projective Implementation). *Let* $\mathcal{P} = (P, Q)$ *be a binary outcome POVM. Let* $\mathcal{D}$ *be a finite set of distributions over outcomes* $\{0, 1\}$. *Let* $\mathcal{E} = \{E_D\}_{D\in\mathcal{D}}$ *be a projective measurement with index set* $\mathcal{D}$. *Consider the following measurement:*

- *Measure under the projective measurement* $\mathcal{E}$ *and obtain a distribution* D *over* $\{0, 1\}$;
- *Output a bit according to the distribution* D.

We say the above measurement is a projective implementation of $\mathcal{P}$ *if it is equivalent of* $\mathcal{P}$, *denoted as* $\mathsf{ProjImp}(\mathcal{P})$.

Note that if the outcome is a distribution $D = (d_0, d_1)$, the collapsed state is an eigenvector of P corresponding to eigenvalue d_0, and it is also an eigenvector of Q corresponding to eigenvalue $d_1 = 1 - d_0$.

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

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

Definition 8 (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 $(\mathsf{TI}_\gamma(\mathcal{P}), \mathbf{I} - \mathsf{TI}_\gamma(\mathcal{P}))$:*

- *Instead of measuring under the projective measurement $\mathcal{E} = \{E_D\}_{D\in\mathcal{D}}$ and obtain a distribution D over $\{0,1\}$, $\mathsf{TI}_\gamma(\mathcal{P})$ measures if the corresponding distribution $D = (d_0, d_1)$ has $d_0 \geq \gamma$.*
- *Output 0 with probability $\mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P})\rho]$ and 1 with probability $1 - \mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P})\rho]$, for any quantum state ρ.*

Therefore, $\mathsf{TI}_\gamma(\mathcal{P})$ is a projection and the collapsed state is a (mixed) state in the span of all eigenvectors of P whose eigenvalues are at least γ.

Remark 1. For a binary outcome measurement $\mathcal{P} = (P_0, P_1)$, we usually say 'perform measurement P_0 on ρ' if $\mathcal{P}$ was performed on ρ. Since we only focus on the case that outcome is 0 in the paper, it sometimes also denotes applying $\mathcal{P}$ on ρ conditioned on that the outcome is 0.

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 9 (Shift Distance). *For two distribution 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 10 ((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 $\Pr[|X - Y| \leq \epsilon] \geq 1 - \delta$.*

Theorem 3 (Theorem 2 in [Zha20]). *Let D be any probability distribution 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

3.1 Unlearnability

Definition 11 (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 in λ.

Definition 12 (γ-Goodness Test with respect to f, D). *Let $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ be a quantum program for computing a classical function $f : [N] \to [M]$. Let D be a probability distribution over the input space $[N]$.*

- *Define $(P_{f,x}, Q_{f,x})$ be a projective measurement that computes the quantum program on input x, and checks in superposition that if the quantum circuit outputs correctly. Let $V_{f,x}$ be a projection that checks if in superposition, the first register is equal to $f(x)$. We have $P_{f,x} = V_{f,x}U_{f,x}$ and $Q_{f,x} = \mathbf{I} - P_{f,x}$.*
- *Let $\{P_f, Q_f\}$ be the controlled projection with respect to the distribution D, as defined in Definition 6. Then, let $\{\mathsf{TI}_\gamma(P_f), \mathbf{I} - \mathsf{TI}_\gamma(P_f)\}$ be the Threshold Implementation for P_f with threshold value γ, as defined in Definition 8.*
- *We say a quantum program is tested γ-**good for computing** f **with distribution** D if the projective measurement $\{\mathsf{TI}_\gamma(P_f), \mathbf{I} - \mathsf{TI}_\gamma(P_f)\}$ on ρ_f outputs 0.*

Definition 13 (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 $\mathcal{D} = \{D_f\}$, and an adversary $\mathcal{A}$ is denoted as $\mathsf{LG}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, which consists the following steps:*

1. ***Sampling Phase**: At the beginning of the game, the challenger takes a security parameter λ and samples a function $f \leftarrow \mathcal{F}_\lambda$;*
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 0 if and only if the program is tested to be γ-good with respect to f, D_f.

Definition 14 (Quantum Unlearnability of $\mathcal{F}$ with Testing Distribution $\mathcal{D}$). *A family of functions with respect to $\mathcal{D}$ is called γ quantum unlearnable if for all λ, for any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds:*

$$\Pr\left[b = 0,\, b \leftarrow \mathsf{LG}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)$$

3.2 Generalized Unlearnability

The γ-goodness test for quantum program (Definition 12) captures the intuition that a quantum program's behavior on classical inputs is γ-good comparing to the input-output behavior of f with respect to the input distribution D_f. For cryptographic primitives, as discussed in the introduction, achieving a particular cryptographic functionality does not necessarily mean to have the exact input-output behavior. As an example, to sign a message, there are usually more than one valid signatures and the intended functionality is preserved as long as any valid signature is provided.

For a randomized function f, we denote the input x of f as the real input taken by f as well as random coins used by f.

Definition 15 (Predicate). *A classical predicate $E(P, y_1, \cdots, y_k, r)$ is a binary outcome function that runs a classical program P on a randomly sampled input x to get output z, and outputs $0/1$ depending on whether $(z, y_1, \cdots, y_k, r) \in R$ for some binary relation defined by R. The randomness of input x, program P all depends on randomness r. $y_1, \cdots, y_k$ are auxiliary inputs that specify the relation.*

Quantumly, it runs a quantum program on random classical input x and measure if $(z, y_1, \cdots, y_n, r) \in R$ in superposition, where z is the first register of the resulting state. In other words, it is a projective measurement indexed by r.

We use $\mathsf{Samp}, \mathscr{F}$ to denote a cryptographic application. $\mathscr{F}$ denotes the intended functionality that this cryptographic application should achieve.

Definition 16 (Cryptographic Application $\mathsf{Samp}, \mathscr{F}$). Samp *is a sampler that takes a security parameter λ and interacts with an adversary $\mathcal{A}$: $f \leftarrow (\mathcal{A} \Leftrightarrow \mathsf{Samp}(1^\lambda))$ where f is a classical circuit that contains some secret information s_f which is unknown to $\mathcal{A}$, and $\mathcal{A}$ can get some public information aux_f from the interaction.*

$\mathscr{F} = \{F_\lambda\}$ and $F_\lambda(P, f, r)$ is a predicate which takes a program, a circuit f and randomness r. For all efficient $\mathcal{A}$, all f sampled by Samp, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, $\Pr[F_\lambda(f, f, r) = 0] \geq 1 - \mathsf{negl}(\lambda)$.

This security of the cryptographic application is orthogonal to its correctness and unlearnability. The definition of security varies a lot when different applications are given. Some examples include CPA security for public key encryption schemes and signature unforgeability. However, the security should be easy to prove, when we implement a copy protection/copy detection scheme using our construction. In this paper, we only focus on its correctness and copy-protect security/copy-detect security/unlearnability/unremovability.

Definition 17 (γ-Goodness Test with respect to f, E). *Let a quantum program for computing f be $(\rho_f, \{U_{f,x}\}_{x \in [N]})$.*

- *Quantumly, define $(P_{f,r}, Q_{f,r})$ be a projective measurement that computes the quantum program on input x (sampled according to r), and checks in superposition that if the output of the quantum circuit satisfies the predicate $E(\cdot, f, r)$ in superposition.*

– *Let* $\{P_f, Q_f\}$ *be the controlled projection with respect to uniform distribution on randomness* r*. Let* $\{\mathsf{TI}_\gamma(P_f), \mathbf{I} - \mathsf{TI}_\gamma(P_f)\}$ *be the threshold implementation for* P_f *with threshold value* γ*.*
– *A quantum program is* ***tested*** γ***-good*** *with respect to* f, E *if the projective measurement* $\{\mathsf{TI}_\gamma(P_f), \mathbf{I} - \mathsf{TI}_\gamma(P_f)\}$ *on* ρ_f *outputs 0.*

Note that Definition 12 fits into this general definition, where the predicate E on a random input x (x is drawn depending on randomness r) and f, checks if the output is equal to $f(x)$.

We then generalize the learning game to the setting of cryptographic applications. Note that $\mathscr{E}$ may be not the same as $\mathscr{F}$. In the game below, an adversary tries to learn a more restricted functionality of f.

Definition 18 (Learning Game for $\mathsf{Samp}, \mathscr{E}$**).** *A learning game for a sampler* Samp *(which samples a function in* $\mathcal{F}_\lambda$*), a predicate* $\mathscr{E} = \{E_\lambda\}$*, and an adversary* $\mathcal{A}$ *is denoted as* $\mathsf{LG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$*, which consists the following steps:*

1. ***Sampling Phase****: At the beginning of the game,* $\mathcal{A}$ *interacts with the challenger and samples* $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$*.*
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 0 *if and only if the program is tested to be* γ*-good with respect to* f, E_λ*.*

It is easy to see that Definition 18 implies Definition 13. One example is digital signature. Samp picks a pair of signing key and verification key $(\mathsf{sk}, \mathsf{vk})$ and outputs a signing circuit $f = \mathsf{Sign}(\mathsf{sk}, \cdot)$ which hard-wires sk and appends vk with the circuit description. The predicate is defined as: sample m, r_s, r_v according to randomness r, run the program with input m and randomness r_s to obtain outcome z, decode sk, vk from the circuit f and the predicate is 0 if and only if $\mathsf{Ver}(\mathsf{vk}, m, z; r_v) = 1$. In other words, the predicate checks if the program outputs a valid signature on a random message.

Definition 19 (Quantum Unlearnability of $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$**).** $((\mathsf{Samp}, \mathscr{F}), \mathscr{E})$ *is called* γ*-quantum-unlearnable if for all* λ*, for any QPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\Pr\left[b = 0,\, b \leftarrow \mathsf{LG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda)$$

3.3 Generalized Copy Protection

Definition 20 (Quantum Copy Protection). *A quantum copy-protection scheme for* $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ *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 *and measuring the first register of the state.*

Efficiency: Setup, Compute and Generate should run in $\mathsf{poly}(\lambda)$ time.

Correctness: For all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that,

- **unique output**: 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 $z_{f,x}$;
- **functionality preserving**: $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ are $(1 - \mathsf{negl}(\lambda))$-good with respect to f, F_λ with probability 1.

Security: It has γ-anti-piracy security defined below.

Note that the property "unique output" enables the copy-protected program can be evaluated polynomially many times.

Definition 21 (γ-Anti-Piracy Security Game). *An anti-piracy security game for a sampler* Samp*, a predicate* $\mathscr{E}$ *and adversary* $\mathcal{A}$ *is denoted as* $\mathsf{AG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$*, which consists of the following steps:*

1. ***Setup Phase****: At the beginning of the game, the challenger takes a security parameter* λ *and obtains secret key* $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. ***Sampling Phase****:* $\mathcal{A}$ *interacts with the challenger and samples* $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$.
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* $(\{U_{R_1,x}\}_{x\in[N]}, \{U_{R_2,x}\}_{x\in[N]})$ *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 0 *if and only if* both *programs* $\mathsf{P}_1, \mathsf{P}_2$ *are both tested to be* γ*-good with respect to* E_λ.

Similarly, we can define q-collusion resistant γ-anti-piracy security game $\mathsf{AG}^{q,\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, in which the adversary $\mathcal{A}$ can make at most q queries in the query phases and is required to output $q+1$ programs $\{(\sigma[R_i], \{U_{R_i,x}\}_{x\in[N]})\}_{i\in[q+1]}$ such that each program is tested to be γ-good.

Definition 22 (γ-Anti-Piracy-Security). *A copy protection scheme for* Samp *and* $\mathscr{E}$ *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* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[b = 0, b \leftarrow \mathsf{AG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{1}$$

3.4 Generalized Copy Detection

A copy detection scheme for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ 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 23 (Quantum Copy Detection). *A quantum copy-detection scheme for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ 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 20.*

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

Correctness ($\mathsf{Generate}$): The same as the security of Definition 20.

Correctness (Check): For all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$, all $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{Generate}(\mathsf{sk}, f)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that, $\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, \rho_f, \{U_{f,x}\}_{x\in[N]})$ outputs 0 with probability at least $1 - \mathsf{negl}(\lambda)$.

Security: It has γ-copy-detection security defined below.

Definition 24 (γ-Copy-Detection Security Game). *A copy-detection security game for a sampler* Samp*, a predicate $\mathscr{E}$ and adversary $\mathcal{A}$ is denoted as $\mathsf{DG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, 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**: $\mathcal{A}$ interacts with the challenger and samples $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$. Let aux_f denote the public information $\mathcal{A}$ obtains during the interaction.*
3. ***Query Phase**: $\mathcal{A}$ makes a single query to the challenger and obtains a copy detection 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 state σ over two registers R_1, R_2 and two sets of unitaries $(\{U_{R_1,x}\}_{x\in[N]}, \{U_{R_2,x}\}_{x\in[N]})$. 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 0 *if and only if*

- *Apply* Check *on input* $\mathsf{pk}, \mathsf{aux}_f, P_i$ *respectively and both outcomes are* 0*. Let P'_i be the collapsed program conditioned on outcomes are* 0*.*
- Both *programs P'_1, P'_2 are both tested to be γ-good with respect to f, E_λ.*

Similarly, we can define q-collusion resistant γ-copy-detection security game $\mathsf{DG}^{\mathcal{A},q}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)$, in which the adversary $\mathcal{A}$ can perform at most q query phases and output $q+1$ programs $P_i = (\sigma[R_i], \{U_{R_i,x}\}_{x\in[N]})$ for $i \in [q+1]$. The game outputs 0 if and only if for all $i \in [q+1]$, the outcome of applying Check on P_i is 0, and the collapsed program P'_i is tested to be γ-good.

Definition 25 (γ-Copy-Detection-Security). *A copy detection scheme for* Samp *and* $\mathscr{E}$ *has* γ*-security, if for any QPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[b = 0, b \leftarrow \mathsf{DG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}(1^\lambda)\right] \leq \mathsf{negl}(\lambda) \tag{2}$$

3.5 Watermarking Primitives with Public Extraction

In this subsection, we give a unified definition that covers most of the definitions in the previous works about watermarking primitives including [CHN+18, KW17, QWZ18, KW19, GKM+19]. We will give several concrete examples of watermarking schemes in Appendix C.

Definition 26 (Watermarking Primitives for $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$). *A watermarking scheme for* $(\mathsf{Samp}, \mathscr{F}), \mathscr{E}$ *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 publicly extractable watermarking scheme. Thus* xk *is always public.*

$\mathsf{Samp}(1^\lambda)$*: it takes a security parameter* 1^λ*,*

$$f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda)).$$

We also denote aux_f *as the public information* $\mathcal{A}$ *obtains during the interaction.*

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

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

Remark. In some watermarking schemes, Setup also outputs a watermarking public parameter wpp and Samp takes this parameter to sample a function. Our construction works in this setting. In sake of clarity, we use the above notion. $\mathsf{Extract}$ may also take an aux that specifies its restricted functionality that f' should achieve. We assume f' contains a piece of information aux as a comment.

It satisfies the following properties.

Definition 27 (Correctness of Mark (Functionality Preserving)). *For all* λ*, 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* $\tau \in \mathcal{M}_\lambda$*,*

$$\Pr\left[F_\lambda(\widetilde{f}, f, r) = 0 \ : \ \begin{smallmatrix} f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda)) \\ \widetilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau) \end{smallmatrix}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 28 (Correctness of $\mathsf{Extract}$). *For all* λ*, 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* $\tau \in \mathcal{M}_\lambda$*, every* aux*,*

$$\Pr\left[\tau \neq \mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, \widetilde{f} || \mathsf{aux}) \ : \ \begin{smallmatrix} f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda)) \\ \widetilde{f} \leftarrow \mathsf{Mark}(\mathsf{mk}, f, \tau) \end{smallmatrix}\right] \leq \mathsf{negl}(\lambda),$$

where aux_f *is the public information given to* $\mathcal{A}$ *and* $\widetilde{f}||\mathsf{aux}$ *is the program appended with* aux.

Definition 29 (Meaningfulness). *For all* λ*, for every efficient algorithm* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$*, for every* aux*,*

$$\Pr\left[\bot \neq \mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, f||\mathsf{aux}) \ : \ \begin{smallmatrix}(\mathsf{xk},\mathsf{mk})\leftarrow\mathsf{Setup}(1^\lambda)\\ f\leftarrow(\mathcal{A}\Longleftrightarrow\mathsf{Samp}(1^\lambda))\end{smallmatrix}\right] \leq \mathsf{negl}(\lambda).$$

where aux_f *is the public information given to* $\mathcal{A}$ *and* $\widetilde{f}||\mathsf{aux}$ *is the program appended with* aux.

Definition 30 (γ**-Unremovability with respect to** $\mathsf{Samp}, \mathscr{E}$**).** *Consider the following game, denoted as* $\mathsf{WG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}$*:*

1. ***Setup****: The challenger samples* $(\mathsf{xk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$*.* $\mathcal{A}$ *then gets* xk*.*
2. ***Sampling Phase****: The challenger interacts with the algorithm* $\mathcal{A}$ *and samples* $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$*.*
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

$$\mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, f^*) \notin Q \ \ \wedge \ \ \Pr_r[E_\lambda(f^*, f, r) = 1] \geq \gamma$$

We say a watermarking scheme has γ*-unremovability respect to* $\mathsf{Samp}, \mathscr{E}$*, if for all QPT* $\mathcal{A}$*, it wins the above game with negligible probability in* λ*. We say it has* q*-collusion resistant* γ*-unremovability if the number of queries made in the query phase is at most* q*.*

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 corollary on approximating threshold implementation:

Corollary 1. *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*. 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 corollary 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 $p-\delta$ outputs outcome 0 and the collapsed state 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 Corollary is in Appendix D.1.

We can also consider approximating the measurements on bipartite (possibly entangled) quantum state. We will prove a similar statement as Corollary 1.

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 Appendix D.2.

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 $\mathcal{D}$ 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 at the end of this section.

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.

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$: 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 $\rho = |A\rangle\langle A|$ and $\{U_x\}_{x \in [N]}$ 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.

5.1 Correctness and Efficiency

Correctness Given $\rho = |A\rangle\langle A|$ and $\{U_x\}_{x\in[N]}$, 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 $|0\rangle|x\rangle|A^\perp\rangle$. It makes an oracle query $\mathcal{O}_2$ on the state $|0\rangle|x\rangle|A^\perp\rangle$, the resulting state is statistically close to $|y_2\rangle|x\rangle|A^\perp\rangle$ where $y_2 = g(x)$.

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

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 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 14, the quantum copy-protection scheme has anti-piracy security against any quantum polynomial-time adversaries. More formally:

Theorem 4 (Main Theorem). *Let $\mathcal{F}$ be a function families that is γ-quantum-unlearnable 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 12, a QPT adversary $\mathcal{A}$ in the anti-piracy security game $\mathsf{AG}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$, 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 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 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 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 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 $(\sigma, U_{R_1}, U_{R_2})$ are both γ-good 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.

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 Corollary 1,

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

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 respect to f, D_f with non-negligible probability. 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 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 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

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

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 0, 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 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.*

By averaging over all randomness, we have the following lemma:

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

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

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

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* $\mathsf{AG}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. *If* $\Pr[\mathsf{E}_2]$ *is non-negligible, there exists an adversary* $\mathcal{A}_2$ *that wins* $\mathsf{LG}^{\mathcal{A}_2}_{\mathcal{F},\mathcal{D},\gamma-2\epsilon}(1^\lambda)$ *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 $\bot$. Similarly, $\mathcal{O}_2(x,v)$ outputs $g(x)$ if $v \in A^\perp \setminus \{0\}$, and otherwise outputs $\bot$ □

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 breaks 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* $\mathsf{AG}^{\mathcal{A}}_{\mathcal{F},\mathcal{D},\gamma}(1^\lambda)$. *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 open problem.

The General Case. The above proof works for the general case, by simply doing the followings: 1. $\mathsf{TI}, \mathsf{ATI}$ are now defined as the (approximated) projective measurement corresponding to the predicate E_λ; 2. In Lemma 5, 6 and 7, the randomness is taken over the general unlearnability game and copy-protection game.

6 Quantum Copy-Detection

6.1 Construction

Now we construct a copy detection scheme for $\mathsf{Samp}, \mathscr{F}, \mathscr{E}$. Let QM and WM be a public key quantum money scheme and a publicly extractable watermarking scheme for $\mathsf{Samp}, \mathscr{F}, \mathscr{E}$, 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.

$\mathsf{Setup}(1^\lambda)$: 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.

$\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, (\rho_f, \{U_{f,x}\}_{x\in[N]}))$:
- 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 $s' = \mathsf{WM.Extract}(\mathsf{pk} = \mathsf{xk}, \mathsf{aux}_f, f')$;
- if all the checks pass, it outputs 0; otherwise, it outputs 1.

Fig. 2. Quantum copy detection scheme.

6.2 Efficiency and Correctness

First, for all $\lambda \in \mathbb{N}$, all efficient $\mathcal{A}$, every $f \leftarrow (\mathcal{A} \Longleftrightarrow \mathsf{Samp}(1^\lambda))$, the program output 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 **unique output** and **functionality preserving** (with respect to $\mathscr{F}$).

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 5. *Assume* QM *is a quantum money scheme and* WM *is a* q*-collusion resistant for* $\mathsf{Samp}, \mathscr{E}$ *with* γ*-unremovability, the above copy-detection scheme for* $\mathsf{Samp}, \mathscr{F}, \mathscr{E}$ *has* q*-collusion resistant* γ*-copy-detection-security.*

Proof. We prove the case for $q = 1$. 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 program output by $\mathcal{A}$ which wins the game $\mathsf{DG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}$.

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 $\mathsf{Check}(\mathsf{pk}, \mathsf{aux}_f, \cdot)$) on both $\sigma[R_1]$ and $\sigma[R_2]$, and both outcomes are 0.
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 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. 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 $\mathsf{WM.Extract}(\mathsf{xk}, \mathsf{aux}_f, \tilde{f}_b) = S_b$ 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 $\mathsf{WM.Extract}(\mathsf{xk}, \mathsf{aux}_f, \tilde{f}_b) = S_b$. 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, E_λ. 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 $b \in \{1, 2\}$, $\Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma$.

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

$$\begin{aligned}
&\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[\forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma\right] \\
=&\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[(E \vee E') \wedge \forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma\right] \\
\leq&\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[E \wedge \forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0] \geq \gamma\right] + \Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}[E']
\end{aligned}$$

Note that the probability is taken over the randomness of $\mathsf{DG}^{\mathcal{A}}_{\mathsf{Samp},\mathscr{E},\gamma}$. 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 $\mathcal{B}$ does the following (it simulates the challenger for the copy-detection scheme):

- It first runs $\mathsf{WM.Setup}(1^\lambda)$ to get xk, mk and let $\mathsf{sk} = \mathsf{mk}$ and $\mathsf{pk} = \mathsf{xk}$.
- It interacts with $\mathcal{A}$ and samples f.
- Instead of sampling a new quantum money state, it uses the state $|\$\rangle$. Let $s = \mathsf{Ver}(|\$\rangle)$ and $\widetilde{f} \leftarrow \mathsf{WM.Mark}(\mathsf{mk}, f, s)$. It gives the instance $\rho_f = (\widetilde{f}, |\$\rangle)$.
- When $\mathcal{A}$ outputs $(\tilde{f}_1, \tilde{f}_2, \sigma)$, $\mathcal{B}$ outputs σ.

Thus $\Pr[E']$ is exact the probability that both verification gives s. □

Claim 2. $\Pr_{(\tilde{f}_1,\tilde{f}_2,\sigma)}\left[E \wedge \forall b, \Pr_r[E_\lambda(\tilde{f}_b, f, r) = 0]\right] \leq \mathit{negl}(\lambda)$.

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 test F. The reduction is the following ($\mathcal{B}$ simulates the challenger for the copy-detection scheme):

- Given $\mathsf{xk}, \mathsf{aux}_f$ in the watermarking security game, $\mathcal{B}$ prepares a quantum money state $|\$\rangle$ with serial number s and gets the marked circuit $\widetilde{f}$ whose marking is s.
- It prepares $\rho_f = (\widetilde{f}, |\$\rangle)$ and feeds it to $\mathcal{A}$.
- When $\mathcal{A}$ outputs outputs $(\tilde{f}_1, \tilde{f}_2, \sigma)$, $\mathcal{B}$ outputs $\tilde{f}_b$ whose mark is not s, i.e, $\mathsf{Extract}(\mathsf{xk}, \mathsf{aux}_f, \tilde{f}_b) \neq s$.

When $\mathcal{A}$ succeeds, $\mathcal{B}$ breaks the security of the watermarking scheme. □

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

It is natural to extend the proof to q-collusion resistance. We put the proof sketch in Appendix D.3.

Combining with the watermarking primitives (see examples in Appendix C), we can get the corresponding copy-detection schemes.

Acknowledgements

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

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.

BGS13. Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In *Annual Cryptology Conference*, pages 344–360. Springer, 2013.

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.

Gav12. D. Gavinsky. Quantum money with classical verification. In *2012 IEEE 27th Conference on Computational Complexity*, pages 42–52, June 2012.

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.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Annual International Cryptology Conference*, pages 39–56. Springer, 2008.

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.

JLS18. Zhengfeng Ji, Yi-Kai Liu, and Fang Song. Pseudorandom quantum states. In *Annual International Cryptology Conference*, pages 126–152. Springer, 2018.

Kan18. Daniel M Kane. Quantum money from modular forms. *arXiv preprint arXiv:1809.05925*, 2018.

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.

LAF+09. Andrew Lutomirski, Scott Aaronson, Edward Farhi, David Gosset, Avinatan Hassidim, Jonathan Kelner, and Peter Shor. Breaking and making quantum money: toward a new quantum cryptographic protocol. *arXiv preprint arXiv:0912.3825*, 2009.

LSZ20. Qipeng Liu, Amit Sahai, and Mark Zhandry. Quantum immune one-time memories, 2020.

NC02. Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

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.

RS19. Roy Radian and Or Sattath. Semi-quantum money. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, page 132–146. Association for Computing Machinery, 2019.

Sho20. Peter Shor. Quantum money based on lattices. In *Simons Institute for the Theory of Computing*. https://simons.berkeley.edu/talks/quantum-money-based-lattices, 2020.

Wie83. Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983.

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.

Zha19. Mark Zhandry. Quantum lightning never strikes the same state twice. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 408–438. Springer, 2019.

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.

A Basics of Quantum Computation and Quantum Information

For completeness, we provide some of the basic definitions of quantum computing and quantum information, for more details see [NC02].

Quantum states Let $\mathscr{H}$ be a finite Hilbert space. Quantum states over $\mathscr{H}$ are positive semi-definite operators from $\mathscr{H}$ to $\mathscr{H}$ with unit trace. These are called density matrices, denoted by ρ or σ in this paper.

A quantum state over $\mathscr{H} = \mathbb{C}^2$ is called qubit, which can be represented by the linear combination of the standard basis $\{|0\rangle, |1\rangle\}$. More generally, a quantum system over $(\mathbb{C}^2)^{\otimes n}$ is called an n-qubit quantum system for $n \in \mathbb{N}_+$.

A pure state can be represented by a unit vector in $\mathbb{C}^n$. The standard basis of the Hilbert space of n-qubit pure states is denoted by $\{|x\rangle\}$, where $x \in \{0,1\}^n$. If a state $|\phi\rangle$ is a linear combination of several $|x\rangle$, we say it is in "superposition".

A mixed state is a collection of pure states $|\phi_i\rangle$ for $i \in [n]$, each with associated probability p_i, with the condition $p_i \in [0,1]$ and $\sum_{i=1}^n p_i = 1$. A mixed state can also be represented by the density matrix: $\rho := \sum_{i=1}^n p_i |\phi_i\rangle \langle\phi_i|$.

Partial Trace. For a quantum state σ over two registers R_1, R_2 (i.e. Hilbert spaces $\mathscr{H}_{R_1}, \mathscr{H}_{R_2}$), we denote the state in R_1 as $\sigma[R_1]$, where $\sigma[R_1] = \mathrm{Tr}_2[\sigma]$ is a partial trace of σ. Similarly, we denote $\sigma[R_2] = \mathrm{Tr}_1[\sigma]$.

Purification of mixed states. For a mixed state ρ over $\mathscr{H}_A$, there exists another space $\mathscr{H}_B$ and a pure state $|\psi\rangle$ over $\mathscr{H}_A \otimes \mathscr{H}_B$ such that ρ is a partial trace of $|\psi\rangle\langle\psi|$ with respect to $\mathscr{H}_B$.

Definition 31 (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*

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

Quantum Measurements In this work, we will use the following general form of measurements.

Definition 32 (Positive operator-valued measure, POVM). *A positive operator-valued measure (POVM) $\mathcal{M}$ is specified by a finite index set $\mathcal{I}$ and a set $\{M_i\}_{i\in\mathcal{I}}$ of Hermitian positive semi-definite matrices M_i such that $\sum_{i\in\mathcal{I}} M_i = \mathbf{I}$.*

When applying $\mathcal{M}$ to a quantum state ρ, the outcome is i with probability $p_i = \mathrm{Tr}[\rho P_i]$ for all $i \in \mathcal{I}$.

To characterize the post-measurement states, we define the quantum measurements as follows.

Definition 33 (Quantum measurement). *A quantum measurement $\mathcal{E}$ is specified by a finite index set $\mathcal{I}$ and a set $\{E_i\}_{i\in\mathcal{I}}$ of measurement operators E_i such that $\sum_{i\in\mathcal{I}} E_i^\dagger E_i = \mathbf{I}$.*

When applying $\mathcal{E}$ to a quantum state ρ, the outcome is i with probability $p_i = \mathrm{Tr}[\rho E_i^\dagger E_i]$ for all $i \in \mathcal{I}$. Furthermore, conditioned on the outcome being i, the post-measurement state is $E_i \rho E_i^\dagger / p_i$.

Note that POVM $\mathcal{M}$ and quantum measurement $\mathcal{E}$ are related by setting $M_i = E_i^\dagger E_i$. In this case, we say that $\mathcal{E}$ is an implementation of $\mathcal{M}$. The implementation of a POVM may not be unique.

Definition 34 (Projective measurement and projective POVM). *A quantum measurement $\mathcal{E}$ is projective if for all $i \in \mathcal{I}$, E_i is a projection, i.e., E_i is Hermitian and $E_i^2 = E_i$.*

Similarly, a POVM $\mathcal{M}$ is projective if each M_i is projection for $i \in \mathcal{I}$.

B Cryptographic Primitives

B.1 Public-key Quantum Money

Definition 35 (Public Key Quantum Money). *A public-key (publicly-verifiable) quantum money consists of the following algorithms:*

- $\mathsf{KeyGen}(1^\lambda)$: *takes as input a security parameter λ, and generates a key pair $(\mathsf{sk}, \mathsf{pk})$.*
- $\mathsf{GenNote}(\mathsf{sk})$: *takes a secret key sk and generates a quantum banknote state $|\$\rangle$.*
- $\mathsf{Ver}(\mathsf{pk}, |\$'\rangle)$: *takes a public key pk, and a claimed money state $|\$'\rangle$, and outputs either 1 for accepting or 0 for rejecting.*

A secure public-key quantum money should satisfy the following properties:

***Verification Correctness**: there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds for any $\lambda \in \mathbb{N}$,*

$$\Pr_{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(1^\lambda)}[\mathsf{Ver}(\mathsf{pk}, \mathsf{GenNote}(\mathsf{sk})) = 1] \geq 1 - \mathsf{negl}(\lambda)$$

***Unclonable Security**: Suppose a QPT adversary is given $q = \mathsf{poly}(\lambda)$ number of valid banknotes $\{\rho_i\}_{i\in[q]}$ and then generates $q' = q+1$ banknotes $\{\rho'_j\}_{j\in[q']}$ where ρ'_j are potentially entangled, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $\lambda \in \mathbb{N}$,*

$$\Pr_{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(1^\lambda)}\left[\forall i \in [q'], \mathsf{Ver}(\mathsf{pk}, \rho'_j) = 1 : \{\rho'_j\} \leftarrow \mathcal{A}(1^\lambda, \{\rho_i\}\right] \leq \mathsf{negl}(\lambda)$$

Remark 2. In rest of the paper, q is set to be 1 for simplicity, and the scheme satisfies unclonable security if $\mathcal{A}$ cannot produce two banknotes that pass verification. [AC12] shows that any public-key quantum money scheme that satisfies security when $q = 1$, can be generalized to a scheme that is secure when $q = \mathsf{poly}(\lambda)$, using quantum-secure digital signatures.

A non-perturb property is also required. That is, one can verify a quantum banknote polynomially many times and the banknote is still a valid banknote. Since Ver is almost a deterministic function, by Gentle Measurement Lemma (Lemma 1), the above definition implies the non-perturb property.

In some settings, instead of outputting 0/1, Ver is required to output either $\perp$ which indicates the verification fails, or a serial number $s \in \mathcal{S}_\lambda$ if it passes the verification. In this case, the scheme should satisfy the following correctness (unique serial number property) and unclonable security [Zha19]:

Unique Serial Number: For a money state $|\$\rangle$, let $H_\infty(|\$\rangle) = -\log \min_s \Pr[\mathsf{Ver}(|\$\rangle) = s]$. We say a quantum scheme has unique serial number property, if $\mathbb{E}[H_\infty(|\$\rangle)]$ is negligible for all λ, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $|\$\rangle$ is sampled from $\mathsf{GenNote}(\mathsf{sk})$.

Unclonable Security: Consider the following game with a challenger and an adversary,

1. The challenger runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $|\$\rangle \leftarrow \mathsf{GenNote}(\mathsf{sk})$, it then runs Ver to get a serial number s.
2. $\mathcal{A}$ is given the public key pk, the banknote $|\$\rangle$ and the serial number s.
3. $\mathcal{A}$ produces σ^* (which contains two separate registers, but they may be entangled) and denotes $\sigma_1 = \mathsf{Tr}_2[\sigma^*]$ and $\sigma_2 = \mathsf{Tr}_1[\sigma^*]$.
4. $\mathcal{A}$ wins if and only if $\mathsf{Ver}(\sigma_1) = \mathsf{Ver}(\sigma_2) = s$.

We say a public key quantum money scheme is secure, if for all QPT $\mathcal{A}$, it wins the above game with negligible probability in λ.

B.2 Obfuscation

Definition 36 (Virtual Black-Box Obfuscation, [BGI+01]). *An obfuscator $\mathcal{O}$ (with auxiliary input) for a collection of circuits $\mathcal{C} = \bigcup_{\lambda\in\mathbb{N}} \mathcal{C}_\lambda$ is a (worst-case) VBB obfuscator if it satisfies:*

- ***Functionality-Preserving:*** *For every $C \in \mathcal{C}$, every input x, $\Pr[\mathcal{O}(C)(x) = C(x)] = 1$.*
- ***Virtual Black-Box:*** *For every poly-size adversary $\mathcal{A}$, there exists a poly-size simulator $\mathcal{S}$, such that for every $\lambda \in \mathbb{N}$, auxiliary input $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, and every predicate $\pi : \mathcal{C}_\lambda \to \{0,1\}$, and every $\mathcal{C} \in \mathcal{C}_\lambda$:*

$$\left| \Pr_{\mathcal{A},\mathcal{O}}[\mathcal{A}(\mathcal{O}(C), \mathsf{aux}) = \pi(C)] - \Pr_{\mathcal{S}}[\mathcal{S}^C(1^\lambda, \mathsf{aux})) = \pi(C)] \right| \leq \mathsf{negl}(\lambda)$$

where the probability is over $C \leftarrow \mathcal{C}_\lambda$, and the randomness of the algorithms $\mathcal{O}, \mathcal{A}$ and $\mathcal{S}$.

C Examples of Watermarking Primitives

Let us look at how the definitions in [CHN+18, GKM+19] fit into our frameworks.

1. Watermarkable PRF in [CHN+18]:
   - $\mathsf{Setup}(1^\lambda) = (\mathsf{wpp}, \mathsf{xk}, \mathsf{mk})$;
   - $\mathsf{Samp}(1^\lambda, \mathsf{wpp})$ samples a PRF key k, $f = \mathsf{PRF}(k, \cdot)$, $\mathsf{aux}_f = \bot$.
   - $F_\lambda(\tilde{f}, f, r)$ is 0 if and only if it samples a random input x (according to r), and $\tilde{f}(x) = f(x)$.
   - Unremovability is defined by $E_\lambda = F_\lambda$. $\gamma = 1/2 + 1/\mathsf{poly}(\lambda)$.
2. Watermarkable signature in [GKM+19]:
   - $\mathsf{Setup}(1^\lambda) = (\mathsf{wpp}, \mathsf{xk}, \mathsf{mk})$;
   - $\mathsf{Samp}(1^\lambda, \mathsf{wpp})$ samples a pair of keys vk, sk and we interpret $f = \mathsf{Sign}(\mathsf{sk}, \cdot)||\mathsf{vk}$, $\mathsf{aux}_f = \mathsf{vk}$.
   - $F_\lambda(\tilde{f}, f, r)$ is 0 if and only if $\mathsf{Ver}(\mathsf{vk}, m, \tilde{f}(m)) = 1$, where vk is decoded from f and m is sampled by r.
   - Unremovability: $E_\lambda = F_\lambda$. γ is inverse polynomial.
3. Watermarkable public key encryption in [GKM+19]:
   - $\mathsf{Setup}(1^\lambda) = (\mathsf{wpp}, \mathsf{xk}, \mathsf{mk})$;
   - $\mathsf{Samp}(1^\lambda, \mathsf{wpp})$ is defined below:
     - It samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKEGen}(1^\lambda, \mathsf{wpp})$;
     - $f = \mathsf{Dec}(\mathsf{sk}, \cdot)||\mathsf{pk}$, $\mathsf{aux}_f = \mathsf{pk}$.
   - $F_\lambda(\tilde{f}, f, r)$ is defined as:
     - Decode pk from f, sample m according to r;
     - Let $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m)$;
     - It outputs 0 if and only if $\tilde{f}(\mathsf{ct}) = m$.
   - Unremovability: $E_\lambda(\tilde{f}, f, r)$ defined as:
     - Decode pk from f, sample b according to r;
     - Decode $\mathsf{aux} = (m_0, m_1)$ from $\tilde{f}$; if $m_0 = m_1$, outputs 1;
     - Let $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m_b)$;
     - It outputs 0 if and only if $\tilde{f}(\mathsf{ct}) = m_b$.

     And, $\gamma = 1/2 + 1/\mathsf{poly}(\lambda)$.

D Missing Details

D.1 Proof of Corollary 1

Corollary 2 (Corollary 1, restated). *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. Then, $\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(P_D) \cdot \rho'] \geq 1 - 2\delta$.*
- *The expected running time is the same as $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$.*

We give the following fact before proving the corollary.

Fact 1. *Let D_0, D_1 be two real-valued probability distributions with shift distance $\Delta^{\epsilon}_{\mathsf{Shift}} = \delta$. Then, we have*

$$\begin{aligned} \Pr[D_0 \geq x - \epsilon] \geq\ & \Pr[D_1 > x] - \delta, \quad \textit{and} \\ \Pr[D_1 \geq x - \epsilon] \geq\ & \Pr[D_0 > x] - \delta \end{aligned}$$

Proof. We prove the first inequality. By the definition of shift distance, we have

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

Then, $\Pr[D_0 \geq x-\epsilon] = 1-\Pr[D_0 \leq x-\epsilon] \geq 1-\Pr[D_1 \leq x]-\delta = \Pr[D_1 \geq x]-\delta$. The second inequality can be proved in a symmetric way. □

Now, we prove the Corollary 1 in below.

Proof. By Theorem 3, we know that there exists an algorithm $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ that approximates the measurement of $\mathsf{ProjImp}(\mathsf{P}_D)$, i.e.,

$$\Delta^{\epsilon}_{\mathsf{Shift}}(\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \leq \delta.$$

In particular, for any pure quantum state $|\psi\rangle$, let D_A be the distribution of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(|\psi\rangle)$ and D_P be the distribution of $\mathsf{ProjImp}(\mathcal{P}_D)(|\psi\rangle)$.

Then, by Fact 1, we have

$$\Pr[D_A \geq \gamma - \epsilon] \geq \Pr[D_P \geq \gamma] - \delta,$$

Hence, by the definition of threshold implementation (Definition 8) and the construction of the algorithm $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$, we can get

$$\mathrm{Tr}\left[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} |\psi\rangle \langle\psi|\right] \geq \mathrm{Tr}\left[\mathsf{TI}_{\gamma}(\mathcal{P}_D) |\psi\rangle \langle\psi|\right] - \delta.$$

Note that mixed state is just a convex combination of pure states. Hence, by the linearity of trace, for any mixed state ρ, we have

$$\mathrm{Tr}\left[\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma-\epsilon} \cdot \rho\right] \geq \mathrm{Tr}\left[\mathsf{TI}_{\gamma}(\mathcal{P}_D) \cdot \rho\right] - \delta,$$

which proves the first bullet. The second bullet follows the same idea by symmetry.

For the third bullet, notice that the measurement algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}$ and $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ do the same thing to the quantum state. So, ρ' is also the collapsed state after the measurement of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho)$.

Since we assume that the outcome of $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P},D,\gamma}(\rho)$ is 0, it implies the corresponding outcome of $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho)$ is at least γ.

By Theorem 3, $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ is (ϵ,δ)-almost projective, which means that if we apply $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ (again) to ρ', the outcome satisfies

$$\Pr[\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho') < \gamma - \epsilon] < \delta.$$

Theorem 3 also provides that the shift distance between $\mathsf{ProjImp}$ and API is small, which means

$$\Pr[\mathsf{ProjImp}(\mathcal{P}_D)(\rho') \leq \gamma - 2\epsilon] \leq \Pr[\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}(\rho') < \gamma - 2\epsilon + \epsilon] + \delta \leq 2\delta.$$

Hence,

$$\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon} \cdot \rho'] = 1 - \Pr[\mathsf{ProjImp}(\mathcal{P}_D)(\rho') \leq \gamma - 2\epsilon] \geq 1 - 2\delta.$$

The third bullet easily follows from the construction. □

D.2 Proof of Lemma 3

Lemma 8 (Lemma 3, restated). *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,*

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

Proof. We use the hybrid argument to show that $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon} \otimes \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}$ approximates $\mathsf{TI}_\gamma(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_\gamma(\mathcal{P}_{D_2})$.

For brevity, let ATI_1 denote $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma-\epsilon}$ and ATI_2 denote $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma-\epsilon}$. Similarly, let TI_1 denote $\mathsf{TI}_\gamma(\mathcal{P}_{D_1})$ and TI_2 denote $\mathsf{TI}_\gamma(\mathcal{P}_{D_2})$. We first show that,

$$\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{TI}_2)\rho] \leq \mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta. \tag{3}$$

Note that ρ is a bipartite quantum state in $\mathscr{H}_{\mathcal{L}} \otimes \mathscr{H}_{\mathcal{R}}$. So, we can consider $\mathsf{TI}_1 \otimes \mathsf{TI}_2$ as a measurement performed by two parties $\mathcal{L}$ and $\mathcal{R}$. In this way, we can write the trace as the probability that $\mathcal{L}$ gets outcome 0 and $\mathcal{R}$ gets outcome 0:

$$\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{TI}_2)\rho] = \Pr[\mathcal{L} \leftarrow 0 \wedge \mathcal{R} \leftarrow 0].$$

We can see that from $\mathsf{TI}_1 \otimes \mathsf{TI}_2$ to $\mathsf{TI}_1 \otimes \mathsf{ATI}_2$, $\mathcal{L}$ performs the same measurement. Hence, we can condition on the event that $\mathcal{L}$ gets outcome 0 and let ρ_1 be the remaining mixed state that traced out the $\mathcal{L}$-part. Then, we get that

$$\begin{aligned}
\Pr[\mathcal{L} \leftarrow 0 \wedge R \leftarrow 0] &= \Pr[\mathcal{L} \leftarrow 0] \cdot \Pr[R \leftarrow 0 | \mathcal{L} \leftarrow 0] \\
&= \Pr[\mathcal{L} \leftarrow 0] \cdot \mathrm{Tr}[\mathsf{TI}_2 \cdot \rho_1] \\
&\leq \Pr[\mathcal{L} \leftarrow 0] \cdot (\mathrm{Tr}[\mathsf{ATI}_2 \cdot \rho_1] + \delta) \\
&\leq \mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta,
\end{aligned}$$

where the first inequality follows from Corollary 1 and the last step follows from $\Pr[\mathcal{L} \leftarrow 0] \leq 1$.

The next step is to show that:

$$\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] \leq \mathrm{Tr}[(\mathsf{ATI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta. \tag{4}$$

In this case, $\mathcal{R}$ performs the same measurement. We can condition on the event that $\mathcal{R}$ gets outcome 0 and let ρ_2 be the remaining mixed state traced out the $\mathcal{R}$-part.

Hence, by a similar argument, we get that

$$\begin{aligned}
\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] &= \Pr[\mathcal{L} \leftarrow 0 \wedge \mathcal{R} \leftarrow 0] \\
&= \Pr[\mathcal{R} \leftarrow 0] \cdot \Pr[\mathcal{L} \leftarrow 0 | \mathcal{R} \leftarrow 0] \\
&= \Pr[\mathcal{R} \leftarrow 0] \cdot \mathrm{Tr}[\mathsf{TI}_1 \cdot \rho_2] \\
&\leq \Pr[\mathcal{R} \leftarrow 0] \cdot (\mathrm{Tr}[\mathsf{ATI}_1 \cdot \rho_2] + \delta) \\
&\leq \mathrm{Tr}[(\mathsf{ATI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta.
\end{aligned}$$

Combining the Eq. (3) and Eq. (4) proves the first bullet of the lemma:

$$\begin{aligned}
\mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{TI}_2)\rho] &\leq \mathrm{Tr}[(\mathsf{TI}_1 \otimes \mathsf{ATI}_2)\rho] + \delta \\
&\leq \mathrm{Tr}[(\mathsf{ATI}_1 \otimes \mathsf{ATI}_2)\rho] + 2\delta.
\end{aligned}$$

For the second part of the lemma, the trace can also be written as

$$\begin{aligned}
\mathrm{Tr}\left[\left(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})\right)\rho'\right] &= \Pr[\mathcal{L} \leftarrow 0 \wedge \mathcal{R} \leftarrow 0] \\
&= \Pr[\mathcal{L} \leftarrow 0] \cdot \Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0],
\end{aligned}$$

where $\mathcal{L}$ and $\mathcal{R}$ are now performing measurements on ρ'.

We first rewrite the term $\Pr[\mathcal{L} \leftarrow 0]$ as

$$\Pr[\mathcal{L} \leftarrow 0] = \mathrm{Tr}[(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathbf{I})\rho'].$$

We can see that this measure process is equivalent to the following process:

1. $\mathcal{R}$ first performs the measurement $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on the $\mathcal{R}$-part of ρ and gets a state ρ_1 such that $\mathrm{Tr}_{\mathcal{L}}[\rho_1] = \mathrm{Tr}_{\mathcal{L}}[\rho']$.
2. $\mathcal{L}$ measures $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ on $\mathrm{Tr}_{\mathcal{R}}[\rho_1]$ and get the collapsed state ρ_2 such that $\rho_2 = \mathrm{Tr}_{\mathcal{R}}[\rho']$.

3. $\mathcal{L}$ measures $\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1})$ on ρ_2.

Hence, we have

$$\mathrm{Tr}[(\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \otimes \mathbf{I})\rho'] = \mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \cdot \rho_2],$$

By Corollary 1 (the third bullet),

$$\mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1}) \cdot \rho_2] \geq 1 - 2\delta.$$

Hence, we get that $\Pr[\mathcal{L} \leftarrow 0] \geq 1 - 2\delta$.

For the second term $\Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0]$, it can be written as

$$\Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0] = \mathrm{Tr}[(\mathbf{I} \otimes \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})) \cdot \rho_3] = \mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2}) \cdot \rho_4],$$

where ρ_3 is the collapsed state conditioned on the outcome of $\mathcal{L}$ being 0 and $\rho_4 = \mathrm{Tr}_{\mathcal{L}}[\rho_3]$.

This measure process is equivalent to the followings:

1. $\mathcal{L}$ first performs two consecutive measurements $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_1,D_1,\gamma}$ and $\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_1})$ on the $\mathcal{L}$-part of ρ, and gets the collapsed state ρ'' such that $\mathrm{Tr}_{\mathcal{R}}[\rho''] = \mathrm{Tr}_{\mathcal{R}}[\rho_3]$.
2. $\mathcal{R}$ measures $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_2,D_2,\gamma}$ on $\mathrm{Tr}_{\mathcal{L}}[\rho'']$ and gets ρ_3.
3. $\mathcal{R}$ measures $\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2})$ on ρ_4.

By Corollary 1 again, we have

$$\Pr[\mathcal{R} \leftarrow 0 | \mathcal{L} \leftarrow 0] = \mathrm{Tr}[\mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_2}) \cdot \rho_4] \geq 1 - 2\delta.$$

Therefore, we have

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

which completes the proof of the second part of the lemma. □

Notice that Lemma 3 can be easily generalized to the case of q-partite state. We state the following corollary without proof:

Corollary 3. *Let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_q$ be q collections of projective measurements and D_i be any probability distributions defined on the index set of $\mathcal{P}_i$ for all $i \in [q]$. For any $0 < \epsilon, \delta, \gamma < 1$, for all $i \in [q]$, the algorithms $\mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_i,D_i,\gamma}$ satisfy the followings:*

- *For any q-partite (possibly entangled, mixed) quantum state $\rho \in \mathscr{H}_1 \otimes \cdots \otimes \mathscr{H}_q$,*

$$\mathrm{Tr}\left[\left(\bigotimes_{i=1}^{q} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_i,D_i,\gamma-\epsilon}\right)\rho\right] \geq \mathrm{Tr}\left[\left(\bigotimes_{i=1}^{q} \mathsf{TI}_{\gamma}(\mathcal{P}_{D_i})\right)\rho\right] - q\delta.$$

- *For any (possibly entangled, mixed) quantum state ρ, let ρ' be the collapsed state after applying $\bigotimes_{i=1}^{q} \mathsf{ATI}^{\epsilon,\delta}_{\mathcal{P}_i,D_i,\gamma-\epsilon}$ on ρ (and normalized). Then,*

$$\mathrm{Tr}\left[\left(\bigotimes_{i=1}^{q} \mathsf{TI}_{\gamma-2\epsilon}(\mathcal{P}_{D_i})\right)\rho'\right] \geq 1 - 2q\delta.$$

D.3 Proof Sketch of Theorem 5

We briefly sketch the proof for q-collusion resistance which is very similar to the case $q = 1$. Let $(\tilde{f}_1, \cdots, \tilde{f}_{q+1}, \sigma)$ be the output of the adversary. Let $s_1, \cdots, s_q$ be the serial numbers in the $\mathsf{Generate}$ procedure. Let $s'_1, \cdots, s'_{q+1}$ be the serial numbers corresponding to σ. If $\mathcal{A}$ succeeds, there are two cases:

1. $\{s'_i\}_{i\in[q+1]} \subseteq \{s_j\}_{j\in[q]}$: in this case, $\mathcal{A}$ successfully copies one of the money state. Thus, we can use $\mathcal{A}$ to construct an adversary for the quantum money scheme.
2. $\{s'_i\}_{i\in[q+1]} \subsetneq \{s_j\}_{j\in[q]}$: in this case, $\mathcal{A}$ successfully unmarks one of the marked program. Thus, we can use $\mathcal{A}$ to construct an adversary for the watermarking scheme.

Therefore, assuming the existence of q-collusion resistant quantum money scheme and watermarking scheme, the construction above is a q-collusion resistant copy-detection scheme.

E Public-key Quantum Money from Copy Detection

In this section, we show that we can use quantum copy detection and public-key encryption to construct a public-key quantum money scheme. This implication shows one more application of copy detection and further demonstrates the relationship between copy detection and public-key quantum money.

We give the following the construction of the public-key quantum money. Assume that we have an underlying public key encryption scheme called $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Enc})$ with message space $\mathcal{M}$, and an underlying copy detection scheme $\mathsf{CD} = (\mathsf{CD.Setup}, \mathsf{CD.Generate}, \mathsf{CD.Compute}, \mathsf{CD.Check})$.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:

- Take in security parameter λ
- Run $\mathsf{PKE.KeyGen}(1^\lambda) \to (\mathsf{PKE.pk}, \mathsf{PKE.sk})$ and $\mathsf{CD.Setup}(1^\lambda) \to (\mathsf{CD.pk}, \mathsf{CD.sk})$.
- Output $\mathsf{pk} = (\mathsf{PKE.pk}, \mathsf{CD.pk})$ and $\mathsf{sk} = (\mathsf{PKE.sk}, \mathsf{CD.sk})$.

$\mathsf{GenNote}(\mathsf{sk}) \to |\$\rangle$:

- Take in the secret key $\mathsf{sk} = (\mathsf{PKE.sk}, \mathsf{CD.sk})$.
- Run $\mathsf{CD.Generate}(\mathsf{CD.sk}, f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$ to generate a copy detection program $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ for the function $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$.
- Output $|\$\rangle = (\rho_f, \{U_{f,x}\}_{x\in[N]})$.

$\mathsf{Ver}(\mathsf{pk}, |\$'\rangle) \to 0/1$:

- Take in the public key $\mathsf{pk} = (\mathsf{PKE.pk}, \mathsf{CD.pk})$ and a claimed banknote state $|\$'\rangle$, i.e. a claimed copy detection program for $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$.
- Parse the claimed banknote $|\$'\rangle$ as $(\mathsf{aux}_f, \rho'_f, \{U'_{f,x}\}_{x\in[N]})$.
- Run $\mathsf{CD.Check}(\mathsf{CD.pk}, \mathsf{aux}_f, \rho'_f, \{U'_{f,x}\}_{x\in[N]}) \to b$; if $b = 1$, output 1 (for reject).
- Test if the program $(\rho'_f, \{U'_{f,x}\}_{x\in[N]})$ is a γ-good program with respect to f, E_λ, using the public information in $\mathsf{pk} = (\mathsf{PKE.pk}, \mathsf{CD.pk})$; if yes, output 0; else output 1.

Fig. 3. Public-key Quantum Money Scheme from Copy Detection

Security Analysis We now show that the public-key quantum money construction has correctness and unclonable security, given a quantum copy detection scheme with correctness and γ-anti-piracy security. The proof is intuitive and we omit some details.

Verification Correctness By the computation correctness of the underlying copy detection scheme CD and decryption correctness of the underlying PKE, a valid banknote $|\$\rangle = (\rho_f, \{U_{f,x}\}_{x\in[N]})$ for $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$ is supposed to pass Check and be a γ-good program with respect to f, E_λ with all but negligible probability. Therefore, verification correctness holds.

Unclonable Security We give a brief proof for the unclonable security of the quantum money scheme, whose security definition is given in Definition 35.

Lemma 9. *Assuming that the quantum copy-protection scheme* CD *has* γ*-anti-piracy, then public-key quantum money scheme has unclonable security.*

Proof. Suppose there is a QPT adversary $\mathcal{A}$ that breaks unclonable security, then we can construct a QPT adversary B that breaks γ-anti-piracy security for CD.

The quantum copy detection challenger interacts with B in a copy detection anti-piracy game: In the Setup phase, challenger runs the setup $\mathsf{CD.Setup}(1^\lambda)$ to generate the keys $(\mathsf{CD.pk}, \mathsf{CD.sk})$. In the Sampling phase, the challenger samples $f = \mathsf{PKE.Dec}(\mathsf{PKE.sk}, \cdot)$, where $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$; note that it gives $\mathsf{aux} = \mathsf{PKE.pk}$ to adversary B and $s_f = \mathsf{PKE.sk}$ is kept secret. B then gives $(\mathsf{CD.pk}, \mathsf{PKE.pk})$ to the quantum money adversary $\mathcal{A}$ as the public key. In the Query phase, copy detection challenger generates one copy of copy detection program $(\rho_f, \{U_{f,x}\}_{x\in[N]}) \leftarrow \mathsf{CD.Generate}(\mathsf{CD.sk}, f)$ and gives to B. Then B sends $(\rho_f, \{U_{f,x}\}_{x\in[N]})$ as a money state $|\$\rangle$ to $\mathcal{A}$. Finally, $\mathcal{A}$ output two claimed money states $\{|\$_i\rangle\} = \{\rho_i, U_i\}_{i\in[2]}$ and sends to B. B uses them as its pirate programs and passes to copy detection challenger. It is easy to see that if both claimed money states $\{|\$_i\rangle\}_{i\in[2]}$ produced by $\mathcal{A}$'s pass verification with non-negligible probability, then B wins the copy detection anti-piracy security game with non-negligible probability.