On the (Im)plausibility of Public-Key Quantum Money from Collision-Resistant Hash Functions

Prabhanjan Ananth* UCSB
Zihan Hu† Tsinghua University
Henry Yuen‡ Columbia University

Abstract

Public-key quantum money is a cryptographic proposal for using highly entangled quantum states as currency that is publicly verifiable yet resistant to counterfeiting due to the laws of physics. Despite significant interest, constructing provably secure public-key quantum money schemes based on standard cryptographic assumptions has remained an elusive goal. Even proposing plausibly-secure candidate schemes has been a challenge.

These difficulties call for a deeper and systematic study of the structure of public-key quantum money schemes and the assumptions they can be based on. Motivated by this, we present the first black-box separation of quantum money and cryptographic primitives. Specifically, we show that collision-resistant hash functions cannot be used as a black-box to construct public-key quantum money schemes where the banknote verification makes classical queries to the hash function. Our result involves a novel combination of state synthesis techniques from quantum complexity theory, oracle compilation and simulation techniques, including Zhandry's compressed oracle technique.

1 Introduction

Unclonable cryptography is an emerging area in quantum cryptography that leverages the no-cloning principle of quantum mechanics [WZ82; Die82] to achieve cryptographic primitives that are classically impossible. Over the years, many interesting unclonable primitives have been proposed and studied. These include quantum copy-protection [Aar09], one-time programs [BGS13], secure software leasing [AL21], unclonable encryption [BL20], encryption with certified deletion [BI20], encryption with unclonable decryption keys [GZ20; CLLZ21], and tokenized signatures [BS16].

One of the oldest and (arguably) the most popular unclonable primitives is quantum money, which was first introduced in a seminal work by Wiesner [Wie83]. A quantum money scheme enables a bank to issue digital money, represented as quantum states, to users with each digital money state associated with a serial number. Informally, the security guarantee states that it is computationally infeasible to produce counterfeit digital money states. That is, a malicious user cannot produce two money states associated with the same serial number that are both accepted by a pre-defined verification procedure. There are two notions we can consider here. The first notion is *private-key* quantum money, where the verification procedure is private. That is, in order to check

*prabhanjan@cs.ucsb.edu
†huzh19@mails.tsinghua.edu.cn
‡hyuen@cs.columbia.edu

whether a money state is valid, we need to submit the state to the bank which decides its validity. A more useful notion is *public-key* quantum money, where anyone can verify the validity of bank notes. While private-key money schemes have been extensively studied and numerous constructions, including information-theoretic ones, have been proposed, the same cannot be said for public-key quantum money schemes.

Aaronson and Christiano [AC13] first demonstrated the feasibility of information-theoretically secure public-key quantum money in the oracle model; meaning that all algorithms in the scheme (e.g., the minting and verification algorithms) query a black box oracle during their execution. In the standard model, there are two types of constructions known for building quantum money:

- In the first category, we have constructions borrowing sophisticated tools from different areas of mathematics, such as knot theory [FGH+12], quaternion algebras [KSS21] and lattices [Zha21; KLS22]. The constructions in this category have been susceptible to cryptanalytic attacks as demonstrated by a couple of recent works [Rob21; BDG22]. We are still in the nascent stages of understanding the security of these candidates.

- In the second category, we have constructions based on existing cryptographic primitives. In this category, we have constructions [Zha21; Shm22a; Shm22b] based on indistinguishability obfuscation, first initiated by Zhandry [Zha21].

We focus on the second category. Constructions from existing primitives, especially from those that can be based on well-studied assumptions, would position public-key quantum money on firmer foundations. Unfortunately, existing constructions of indistinguishability obfuscation are either post-quantum **in**secure [AJL+19; JLS21; JLS22] or are based on newly introduced cryptographic assumptions [GP21; BDGM20; WW21; DQV+21] that have been subjected to cryptanalytic attacks [HJL21].

The goal of our work is to understand the feasibility of constructing public-key quantum money from fundamental and well-studied cryptographic primitives. We approach this direction via the lens of black-box separations. Black-box separations have been extensively studied in classical cryptography [Rud91; Sim98; GKM+00; RTV04; BM09; DLMM11; GKLM12; BDV17]. We say that a primitive A cannot be constructed from another primitive B in a black-box manner if there exists a computational world (defined by an oracle) where B exists but A does not. Phrased another way, these separations rule out constructions of primitive A where primitive B is used in a black-box manner. In this case, we say that there is a black-box separation between A and B. Black-box separations have been essential in understanding the relationship between different cryptographic primitives. Perhaps surprisingly, they have also served as a guiding light in designing cryptographic constructions. One such example is the setting of identity-based encryption (IBE). A couple of works [BPR+08; PRV12] demonstrated the difficulty of constructing IBE from the decisional Diffie Hellman (DDH) assumption using a black-box construction which prompted the work of [DG17] who used non-black box techniques to construct IBE from DDH.

1.1 Our Work

Black-Box Separations for Unclonable Cryptography. We initiate the study of black-box separations in unclonable cryptography. In this work, we study a black-box separation between public-key quantum money and (post-quantum secure) collision-resistant hash functions. To the best of our knowledge, our work takes the first step in ruling out certain approaches to constructing public-key quantum money from well-studied assumptions.

Model. Before we go any further, we need to discuss the model in which we prove the black-box separation. We consider two oracles with the first being a random oracle $\mathcal{R}$ (i.e., a uniformly random function) and the second being a PSPACE oracle (i.e., one that can solve PSPACE-complete problems). We investigate the feasibility of quantum money schemes and collision-resistant hash functions in the presence of $\mathcal{R}$ and PSPACE. That is, all the algorithms of the quantum money schemes and also the adversarial entities are given access to the oracles $\mathcal{R}$ and PSPACE.

There are two ways we can model a quantum algorithm to have access to an oracle. The first is *classical access*, where the algorithms in the quantum money scheme can only make classical queries to the oracle; that is, each query to the oracle is measured in the computational basis before forwarding it to the oracle. If an algorithm A has classical access to an oracle, say $\mathcal{U}$, we denote this by $\mathsf{A}^{\mathcal{U}}$. The second is *quantum access*, where the algorithms can make superposition queries. That is, an algorithm can submit a state of the form $\sum_{x,y} \alpha_{x,y} |x\rangle |y\rangle$ to the oracle $\mathcal{O}$ and it receives back $\sum_{x,y} \alpha_{x,y} |x\rangle |\mathcal{O}(x) \oplus y\rangle$. If an algorithm A has quantum access to an oracle $\mathcal{U}$, we denote this by $\mathsf{A}^{|\mathcal{U}\rangle}$.

Our ultimate goal is to obtain black-box separations in the quantum access model, where the algorithms in the quantum money scheme can query oracles in superposition. However, there are two major obstacles to achieving this.

First, analyzing the quantum access model in quantum cryptography has been notoriously challenging. For example, it is not yet known how to generalize to the quantum access setting black-box separations between key agreement protocols – a *classical* cryptographic primitive – and one-way functions [IR90]. Attempts to tackle special cases have already encountered significant barriers [ACC+22], and has connections to long-standing conjectures in quantum query complexity (c.f. the Aaronson-Ambainis conjecture [AA09]).

Second, we have to contend with the difficulty that quantum money is an *inherently quantum* cryptographic primitive. A black-box separation requires designing an adversary that can effectively clone a quantum banknote given a *single* copy of it. Here one encounters problems of a uniquely quantum nature, such as the No-Cloning Theorem [WZ82; Die82] and the fact that measuring the banknote will in general disturb it.

We present partial progress towards the ultimate goal stated above by simplifying the problem and focusing exclusively on this second obstacle: we prove black-box separations where the banknote verification algorithm in the quantum money schemes makes *classical* queries to the random oracle $\mathcal{R}$ (but still can make quantum queries to the PSPACE oracle), and the minting algorithm may still make quantum queries to both $\mathcal{R}$ and PSPACE oracles. As we will see, even this special case of quantum money schemes is already challenging and nontrivial to analyze. We believe that our techniques may ultimately be extendable to the general setting (if there indeed exists a black-box impossibility in the general setting!), where all algorithms can make quantum queries to all oracles, and furthermore help prove black-box separations of other quantum cryptographic primitives.

Main Theorem. We will state our theorem more formally. A quantum money scheme consists of three QPT algorithms, namely $(\mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Ver})$, where KeyGen produces a public key-secret key pair, Mint uses the secret key to produce money states and a serial number associated with money states and finally, Ver determines the validity of money states using the public key. We consider *oracle-aided* quantum money schemes, where these algorithms have access to a random oracle $\mathcal{R}$ and a PSPACE oracle, defined above.

Theorem 1 (Informal)**.** *Any public-key quantum money scheme* $(\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}, \mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}, \mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle})$ *is insecure.*

By insecurity, we mean the following. There exists an quantum polynomial-time adversary (QPT) $\mathcal{A}$ such that $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, given a money state (pk, ρ_s, s), where pk is the public key and s is a serial number, with non-negligible probability, can produce two (possibly entangled) states that both pass the verification checks with respect to the same serial number s. The probability is taken over the randomness of $\mathcal{R}$ and also over the randomness of $\mathsf{KeyGen}, \mathsf{Mint}$ and $\mathcal{A}$. We note that only KeyGen and Mint can have quantum access to $\mathcal{R}$, while Ver only has classical access. On the other hand, even the adversary $\mathcal{A}$ we demonstrate only makes classical queries to $\mathcal{R}$.

Furthermore, we note that the random oracle $\mathcal{R}$ constitutes a collision-resistant hash function against QPT adversaries that can make queries to $(\mathcal{R}, |\mathsf{PSPACE}\rangle)$ [Zha15]. We note that $\mathcal{R}$ still remains collision-resistant even when the adversaries can make *quantum* queries to $\mathcal{R}$, not just classical ones.

Implications. Our main result rules out a class of public-key quantum money constructions that (a) base their security on collision-resistant hash functions, (b) use the hash functions in a black-box way, and (c) where the verification algorithm makes classical queries to the hash function. Clearly, it would be desirable to generalize the result to the case where the verification algorithm can make quantum queries to the hash function. However, there are some conceptual challenges to going beyond classical verification queries (which we discuss in more detail in Section 2.2.3).

The class of quantum money schemes in this hybrid classical-quantum query model is quite interesting on its own and a well-motivated setting. For example, in Zhandry's public-key quantum money scheme [Zha21], the mint procedure only needs classical access to the underlying cryptographic primitives (when the component that uses cryptographic primitives is viewed as a black-box) while the verification procedure makes quantum queries. In the constructions of copy-protection due to Coladangelo et al. [CLLZ21; CMP20], the copy-protection algorithm only makes classical queries to the cryptographic primitives in the case of [CLLZ21] and the random oracle in the case of [CMP20] whereas the evaluation algorithm in both constructions make quantum queries. Finally, in the construction of unclonable encryption in [AKL+22], all the algorithms only make classical queries to the random oracle. Given these constructions, we believe it is important to understand what is feasible or impossible for unclonable cryptosystems in the hybrid classical-quantum query model.

Secondly, we believe that the hybrid classical-quantum query model is useful testbed for developing techniques needed for black-box separations, and for gaining insight into the structure of unclonable cryptographic primitives. Even in this special case, there are a number of technical and conceptual challenges to overcome in order to get our black-box separation of Theorem 1. We believe that the techniques developed in this paper will be a useful starting point for future work in black-box separations in unclonable cryptography.

Other Separations. As a corollary of our main result, we obtain black-box separations between public-key quantum money and many other well-studied cryptographic primitives such as one-way functions, private-key encryption and digital signatures.

Our result also gives a separation between public-key quantum money and collapsing hash functions in the same setting as above; that is, when Ver makes classical queries to $\mathcal{R}$. This follows from a result due to Unruh [Unr16] who showed that random oracles are collapsing. Collapsing

hash functions are the quantum analogue of collision-resistant hash functions. Informally speaking, a hash function is collapsing if an adversary cannot distinguish a uniform superposition of inputs, say $|\psi\rangle$, mapping to a random output y versus a computational basis state obtained by measuring $|\psi\rangle$ in the computational basis. Zhandry [Zha21] showed that hash functions that are collision-resistant but not collapsing imply the existence of public-key quantum money. Thus our result rules out a class of constructions of quantum money from collapsing functions, improving our understanding of the relationship between them.

2 Our Techniques in a Nutshell

We present a high level overview of the techniques involved in proving Theorem 1. But first, we will briefly discuss the correctness guarantee of *oracle-aided* public-key quantum money schemes.

Reusability. In a quantum money scheme $(\mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Ver})$, we require that Ver accepts a state and a serial number produced by Mint with overwhelming probability. However, for all we know, Ver, during the verification process, might destroy the state. In general, a more useful correctness definition is reusability, which states that a money state can be repeatedly verified without losing its validity. In general, one can show that the gentle measurement lemma [Win99] does prove that correctness implies reusability. However, as observed in [AK22], this is not the case when Ver has classical access to an oracle. Specifically, Ver has classical access to $\mathcal{R}$. Hence, we need to explicitly define reusability in this setting. Roughly speaking, we require the following: suppose we execute Ver on a money state $\rho^{(0)}$ produced using Mint and the verification algorithm accepts with probability δ. The residual state is (possibly) a different state $\rho^{(1)}$ which when executed upon by Ver is also accepted with probability close to δ. In fact, even if we run the verification process polynomially many times, the state obtained at the end of the process should still be accepted by Ver with probability close to δ.

2.1 Warmup: Insecurity when $\mathcal{R}$ is absent

Towards developing techniques to prove Theorem 1, let us first tackle a simpler statement. Suppose we have a secure public-key quantum money scheme $(\mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Ver})$. This means that any QPT adversary cannot break the security of this scheme. But what about oracle-aided adversaries? In more detail, we ask the following question: *Does there exist a QPT algorithm, given quantum access to a* PSPACE *oracle, that violates the security of* $(\mathsf{KeyGen}, \mathsf{Mint}, \mathsf{Ver})$*?* That is, given (s, ρ_s), where s is a serial number and ρ_s is a valid banknote produced by Mint, it should be able to produce two states, with respect to the same serial number s, that are both accepted by the verifier.

Even this seemingly simple question seems challenging! Let us understand why. Classical cryptographic primitives (even post-quantum secure ones) such as encryption schemes or digital signatures can be broken by efficient adversaries who have access to even NP oracles. This follows from the fact that we can efficiently reduce the problem of breaking the scheme to the problem of determining membership in a language. For instance, in order to succeed in breaking an encryption scheme, the adversary has to decide whether the instance $(\mathsf{pk}, \mathsf{ct}, \mathsf{m}) \in L$, where pk is a public key, ct is a ciphertext, m is a message and L consists of instances of the form $(\mathsf{pk}, \mathsf{ct}, \mathsf{m})$, where ct is an encryption of m with respect to the public key pk. Implicitly, we are using the fact that $\mathsf{pk}, \mathsf{ct}, \mathsf{m}$ are binary strings. Emulating a similar approach in the case of quantum money would result in

quantum instances and it is not clear how to leverage PSPACE, or more generally a decider for any language, to complete the reduction.

Synthesizing Witness States. Towards addressing the above question, we reduce the task of breaking the security of the quantum money scheme using PSPACE to the task of finding states accepted by the verifier in quantum polynomial space. This reduction is enabled by the following observation, due to Rosenthal and Yuen [RY21]: a set of pure states computable by a quantum polynomial space algorithm (which may in general include intermediate measurements) can be synthesized by a QPT algorithm with quantum access to a PSPACE oracle. Implicit in the result of [RY21] is the following important point: in order to synthesize the state using the PSPACE oracle, it is important that we know the entire description of the quantum polynomial space algorithm generating the pure states.

In more detail, we show the following statement: for every[1] verification key pk, serial number s, there exists a pure state[2] $\rho_{\mathsf{pk},s}$ that is accepted by $\mathsf{Ver}(\mathsf{pk}, s, \cdot)$ with non-negligible probability and moreover, can be generated by a quantum polynomial space algorithm.

The first attempt is to follow the classical brute-force search algorithm. Namely, we repeat the following for exponential times: guess a quantum state ρ uniformly at random and if ρ is accepted by $\mathsf{Ver}(\mathsf{pk}, s, \cdot)$ with non-negligible probability, output ρ and terminate. (Output an arbitrary state if we run out of times.) However, there are two problems with this attempt. Firstly, in general, it's not clear how to calculate the acceptance probability of $\mathsf{Ver}(\mathsf{pk}, s, \rho)$ in polynomial space (ρ needs exponential bits to represent). Secondly, ρ might be destroyed when we calculate the acceptance probability.

To fix the first problem, we note that an estimation of the acceptance probability is already good enough and it can be done by using an excellent method introduced by Marriott and Watrous [MW05] (called MW technique). To be more specific, MW technique allows us to estimate efficiently the acceptance probability of a verification algorithm on a state with only one copy of that state. Furthermore, it won't destroy the state too much in the sense that the expected acceptance probability of the residual state won't decay, which fixes the second problem. Finally, it is worth noting that the maximally mixed state can be seen as sampling one copy of quantum state uniformly at random, i.e. $\mathbf{E}_{\rho}[\rho] = \frac{1}{2^n} I$ where n is the number of qubits in ρ.

This brings us to our second attempt. We repeat the following process for exponentially many times: apply MW technique on a maximally mixed state and if the estimated acceptance probability happens to be non-negligible, output the residual state and terminate. (Output an arbitrary state if all the iterations fail.)

As the MW technique is efficient, this algorithm only uses polynomial space. Furthermore, intuitively we can get a state that is accepted by Ver with non-negligible acceptance probability given the fact that such a state exists. Because if such state exists, by a simple convexity argument, we can assume that without loss of generality that it's pure. Maximally mixed state can be treated as a uniform mixture of a basis containing that pure state. Thus roughly speaking, we start from that pure state with inverse exponential probability, so we can find it in exponentially many iterations with overwhelming probability. This attempt almost succeeds except that it outputs a *mixed* state in general, but the known approach in [RY21] can only deal with pure states. There are two reasons for this. Firstly, we start with a maximally mixed state and secondly, MW technique involves

[1] Technically, we show a weaker statement which holds for an overwhelming fraction of (pk, s).

[2] Technically, we require that the reduced density matrix of $\rho_{\mathsf{pk},s}$ is accepted by Ver.

intermediate measurements.

Our final attempt makes the following minor changes compared to the second attempt. To fix the first issue, it starts with a maximally entangled state (instead of maximally mixed state) and only operates on half of it. To fix the second issue, it runs the MW process coherently by deferring all the intermediate measurements. Then we will end up with a pure state whose reduced density matrix is the same as the output state of the second attempt.

2.2 Insecurity in the presence of $\mathcal{R}$

So far, we considered the task of violating the security of a quantum money scheme where the algorithms did not have access to any oracle. Let us go back to the oracle-aided quantum money schemes, where, all the algorithms (honest and adversarial) have access to $\mathcal{R}$ and $|\mathsf{PSPACE}\rangle$. Our goal is to construct an adversary that violates the security of quantum money schemes. *But didn't we just solve this problem?* Recall that when invoking [RY21], it was crucial that we knew the entire description of the polynomial space algorithm in order to synthesize the state. However, when we are considering oracle-aided verification algorithms, denoted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, we don't have the full description of[3] $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$. Thus, we cannot carry out the synthesizing process.

A naive approach is to sample our own oracle $\mathcal{R}'$ and synthesize the state with respect to $\mathsf{Ver}^{\mathcal{R}',|\mathsf{PSPACE}\rangle}$. However, this does not help. Firstly, there is no guarantee that $\mathsf{Ver}^{\mathcal{R}',|\mathsf{PSPACE}\rangle}(\mathsf{pk}, s, \cdot)$ accepts any state with high enough probability. Without this guarantee, the synthesizing process does not work. For now, let us take for granted that there does exist some witness state σ' that is accepted by $\mathsf{Ver}^{\mathcal{R}',|\mathsf{PSPACE}\rangle}(\mathsf{pk}, s, \cdot)$ with high enough probability. However, there is no guarantee that σ' is going to be accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(\mathsf{pk}, s, \cdot)$ with better than negligible probability.

Towards addressing these hurdles, we first focus on a simple case when KeyGen and Mint make classical queries to $\mathcal{R}$ and we later, focus on the quantum queries case.

2.2.1 KeyGen and Mint: Classical Queries to $\mathcal{R}$

Compiling out $\mathcal{R}$. Suppose we can magically find a database D, using only polynomially many queries to $\mathcal{R}$, such that all the query-answer pairs made by Ver to $\mathcal{R}$ are contained in D. In this case, there is a QPT adversary $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ that given (pk, s, ρ_s), can find two states (s, σ_s') and (s, σ_s'') such that $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ accepts both the states. $\mathcal{A}$ does the following: it first finds the database D and constructs another circuit $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$ such that $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$ runs $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ and when Ver makes a query to $\mathcal{R}$, the query is answered by D. Then, $\mathcal{A}$ synthesizes two states (s, σ_s') and (s, σ_s''), using PSPACE, such that both the states are accepted by $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$. By definition of the database D, these two states are also accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$.

Of course, it is wishful for us to hope that we can find a database D by making only polynomially many queries to $\mathcal{R}$ that is perfectly consistent with the queries made by Ver. Instead, we hope to recover a good enough database D. In more detail, we aim to recover a database D that captures all the relevant queries made by KeyGen and Mint.

Let D_{KeyGen} and D_{Mint} be the collection of query-answer pairs made by KeyGen and Mint respectively. A query made by Ver is called *bad* if this query is in $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}}$ and moreover, this query was not recorded in D. If Ver makes *bad* queries then the answers returned will likely

[3]The fact that we don't have the description of $\mathcal{R}$ is the problem here. If we had a verifier of the form $\mathsf{Ver}^{|\mathsf{PSPACE}\rangle}$, then we could have synthesized the state with access to the PSPACE oracle.

be inconsistent with $\mathcal{R}$ and thus, there is no guarantee that Ver will work. Our hope is that the probability of Ver making bad queries is upper bounded by an inverse polynomial.

Once we have such a database D, by a similar argument, we can conclude that the states synthesized using $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$ are also accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$.

But how do we recover this database D? To see how, we will first focus on a simple case before dealing with the general case.

State-independent database simulation. Note that the queries made by Ver could potentially depend on its input state. For now, we will assume that the distribution of queries made by Ver is independent of the input state. We will deal with the state-dependent query distributions later.

The first attempt to generate D would be to rely upon techniques introduced by Canetti, Kalai and Paneth [CKP15] who, in a different context – that of proving impossibility of obfuscation in the random oracle model – showed how to generate a database that is sufficient to simulate the queries made by the evaluation algorithm. Suppose (s, ρ_s) is the state generated by Mint. Then, run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk, s, \rho_s)$ a fixed polynomially many times, referred to as *test* executions, by querying $\mathcal{R}$. In each execution of $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, record all the queries made by Ver along with their answers. The union of queries made in all the executions of Ver will be assigned the database D. In the context of obfuscation for classical circuits, [CKP15] argue that, except with inverse polynomial probability, the queries made by the evaluation algorithm can be successfully simulated by D. This argument is shown by proving an upper bound on the probability that the evaluation algorithm makes bad queries.

A similar analysis can also be made in our context to argue that D suffices for successful simulation. That is, we can argue that the state we obtain after all the executions of Ver (which could be very different from the state we started off with) can be successfully simulated using D. However, it is crucial for our analysis to go through that D_{Ver} (the query-answer pairs made during Ver) is *independent* of the state input to Ver.

State-dependent database simulation. For all we know, D_{Ver} could indeed depend on the input state. In this case, we can no longer appeal to the argument of [CKP15]. At a high level, the reason is due to the fact that after each execution of Ver, the money state could potentially change and this would affect the distribution of D_{Ver} in the further executions of Ver in such a way that the execution of Ver on the final state (which could be different from the input state in the first execution of Ver) cannot be simulated using the database D.

Instead, we will rely upon a technique due to [AK22], who studied a similar problem in the context of copy-protection. They showed that by randomizing the number of executions, one can argue that the execution of Ver on the state obtained after all the test executions can be successfully simulated using D, except with inverse polynomial probability. That is, suppose the initial state is $(s, \rho_s^{(0)})$ and after running $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, ℓ number of times, let the resulting state be $(s, \rho_s^{(\ell)})$. Let D be as defined before. Then, we have the guarantee that Ver accepts $(s, \rho_s^{(\ell)})$, except with inverse polynomial probability, even when $\mathcal{R}$ is simulated using D.

This suggests the following attack on the quantum money scheme. On input a money state (s, ρ_s), do the following:

- Run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, ℓ times, also referred to as test executions. The number of times we need to run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, namely ℓ, is randomized as per [AK22]. Let D be the set of query-answer

pairs made by Ver to $\mathcal{R}$ during the test executions. Denote $\rho^{(\ell)}$ to be the state obtained after ℓ executions of Ver.

- Let $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$ be the verification circuit as defined earlier.
- Using quantum access to PSPACE, synthesize two states (s,σ'_s) and (s,σ''_s), as per Section 2.1, such that both the states are accepted by $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$.
- Output (s,σ'_s) and (s,σ''_s).

From the witness synthesis method, we have the guarantee that (s,σ'_s) and (s,σ''_s) are both accepted by $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$. However, this is not sufficient to prove that the above attack works. Remember that the adversary is supposed to output two states that are both accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$. Unfortunately, there is no guarantee that $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ accepts these two states. Indeed, both σ'_s and σ''_s could be quite different from $\rho^{(\ell)}$. Hence, the above attack does not work.

Every mistake we make is progress. Let us understand why the above attack does not work. Note that as long as Ver does not make any bad query (i.e., a query in $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}}$ but not contained in D), it cannot distinguish whether its queries are being simulated by D or $\mathcal{R}$. However, when Ver is executed on (s,σ'_s) or (s,σ''_s), we can no longer upper bound the probability that Ver will not make any bad queries.

We modify the above approach as follows: whenever Ver makes bad queries, we can update the database D to contain the bad queries along with the correct answers (i.e., answers generated using $\mathcal{R}$). Once D is updated, we can synthesize two new states using $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$. We repeat this process until we have synthesized two states that are accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$.

Is there any guarantee that this process will stop? Our key insight is that whenever we make a mistake and synthesize states that are not accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ then we necessarily learn a new query in $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}}$ that is not contained in D. Thus, with each mistake, we make progress. Since there are only a polynomial number of queries in $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}}$, we will ultimately end up synthesizing two states that are accepted by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$.

Our Attack. With this modification, we have the following attack. On input a money state (s,ρ_s), do the following:

- *Test phase*: Run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, ℓ times, also referred to as test executions. The number of times we need to run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$, namely ℓ, is randomized as per [AK22]. Let D be the set of query-answer pairs made by Ver to $\mathcal{R}$ during the test executions. Denote $\rho^{(\ell)}$ to be the state obtained after ℓ executions of Ver.
- *Update phase*: Repeat the following polynomially many times. Let $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$ be the verification circuit as defined earlier. Using quantum access to PSPACE, synthesize a state (s,σ_s) as per Section 2.1, such that the state is accepted by $\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}$. Run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ on this state and include any new queries made by Ver to $\mathcal{R}$ in D.
- Let $D_1,\ldots,D_{\mathrm{poly}}$ be the databases obtained after every execution during the update phase.

- Using quantum access to PSPACE, synthesize two states (s, σ'_s) and (s, σ''_s) such that both the states are accepted by $\widehat{\mathsf{Ver}}^{D_j, |\mathsf{PSPACE}\rangle}$ for some randomly chosen j.
- Output (s, σ'_s) and (s, σ''_s).

In the technical sections, we analyze the above attack and prove that it works.

2.2.2 KeyGen and Mint: Quantum Queries to $\mathcal{R}$

The important point to note here is the form of our aforementioned attacker. It only takes advantage of the fact that Ver makes classical queries to $\mathcal{R}$. When KeyGen and Mint make quantum queries to $\mathcal{R}$ while Ver makes classical queries to $\mathcal{R}$, we can still run the attacker. What is left is to show that the same attacker works even when KeyGen and Mint make quantum queries to $\mathcal{R}$.

The main difficulty in carrying out the intuitions in Section 2.2.1 to the case where KeyGen and Mint make quantum queries to $\mathcal{R}$ is that it's difficult to define analogue of D_{KeyGen} and D_{Mint}. To give a flavour of the difficulty, let's first consider two naive attempts.

The first attempt is to define D_{KeyGen} and D_{Mint} to be those query-answer pairs asked (with non-zero amplitudes) during KeyGen and Mint. However, this attempt suffers from the problem that in this way, $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}}$ can have exponential elements. So even if each time we can make progress in the sense that we recover some new elements in $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}}$, there is no guarantee that the update phase will terminate in polynomial time.

The second attempt is to only include queries that are asked "heavily" during KeyGen and Mint. To be more specific, let D_{KeyGen} and D_{Mint} be query-answer pairs asked with inverse polynomial squared amplitudes during KeyGen and Mint. However, with this plausible definition, the claim does not hold that whenever the acceptance probability of $\widehat{\mathsf{Ver}}^{D, |\mathsf{PSPACE}\rangle}$ is far from that of $\mathsf{Ver}^{\mathcal{R}, |\mathsf{PSPACE}\rangle}$, then we can recover a query in $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D$, which is a crucial idea underlying our intuitions in Section 2.2.1. Let us understand why this claim is not true if we adopt this definition of D_{KeyGen} and D_{Mint}.

Consider the following contrived counterexample $(\mathsf{KeyGen}^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}, \mathsf{Mint}^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}, \mathsf{Ver}^{\mathcal{R}, |\mathsf{PSPACE}\rangle})$. Suppose there exists a quantum money scheme $(\mathsf{KeyGen}'^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}, \mathsf{Mint}'^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}, \mathsf{Ver}'^{\mathcal{R}, |\mathsf{PSPACE}\rangle})$. We modify this scheme into $(\mathsf{KeyGen}^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}, \mathsf{Mint}^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}, \mathsf{Ver}^{\mathcal{R}, |\mathsf{PSPACE}\rangle})$ as follows:

- $\mathsf{KeyGen}^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}(1^n)$: outputs the secret key-public key pair of $\mathsf{KeyGen}'^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}$.
- $\mathsf{Mint}^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}(sk)$: takes as input sk, makes quantum query to $\mathcal{R}$ on state $\frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} |s\rangle |0\rangle$ to get a state $\frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} |s\rangle |\mathcal{R}(s)\rangle$, and then measures the first register to get a value s. It also runs $\mathsf{Mint}'^{|\mathcal{R}\rangle, |\mathsf{PSPACE}\rangle}(sk)$ to get a serial number s' along with a state $\rho_{s'}$. It outputs (s, s') as the serial number and $(\mathcal{R}(s), \rho_{s'})$ as the banknote.
- $\mathsf{Ver}^{\mathcal{R}, |\mathsf{PSPACE}\rangle}(pk, ((s, s'), (h, \rho_{s'})))$: takes as input pk and an alleged banknote $((s, s'), (h, \rho_{s'}))$, makes classical query to $\mathcal{R}$ on the input s to get $R(s)$ and checks if $h = \mathcal{R}(s)$. It also checks if $(s', \rho_{s'})$ is a valid money state with respect to $\mathsf{Ver}'^{\mathcal{R}, |\mathsf{PSPACE}\rangle}$. Accepts if and only if both the checks pass.

In the above counterexample, it is possible that there is no query-answer pair that is asked with inverse polynomial squared amplitudes and thus $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} = \emptyset$. At the beginning $D = \emptyset$ because we have not started to record query-answer pairs. In this case, the acceptance probability of

$\widehat{\mathsf{Ver}}^{D,|\mathsf{PSPACE}\rangle}(pk,((s,s'),(\mathcal{R}(s),\rho_{s'})))$ is smaller than or equal to $\frac{1}{2^m}$ where m is the output length of $\mathcal{R}$ on input length n while the acceptance probability of $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,((s,s'),(\mathcal{R}(s),\rho_{s'})))$ is 1 if $(\mathsf{KeyGen}'^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle},\mathsf{Mint}'^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle},\mathsf{Ver}'^{\mathcal{R},|\mathsf{PSPACE}\rangle})$ is (perfectly) correct. However, it's impossible to recover a new query in $D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}}-D$ because it's empty.

Purified View. Our insight is to consider an alternate world called the *purified view*. In this alternate world, we run everything coherently; in more detail, we consider a uniform superposition of $\mathcal{R}$, run Mint, KeyGen and even the attacker coherently (i.e., no intermediate measurements). If the attacker is successful in this alternate world then he is also successful in the real world where $\mathcal{R}$ and the queries made by Ver to $\mathcal{R}$ are measured. We then employ the the compressed oracle technique by Zhandry [Zha18] to coherently recover the database of query-answer pairs recorded during $\mathsf{KeyGen},\mathsf{Mint}$ and relate this with the database recorded during Ver. Using an involved analysis, we then show many of the insights from the case when $\mathsf{KeyGen},\mathsf{Mint}$ make classical queries to $\mathcal{R}$ can be translated to the quantum query setting.

2.2.3 Challenges To Handling Quantum Verification Queries

It is natural to wonder whether we can similarly use the compressed oracle technique to handle quantum queries made by Ver. Unfortunately, there are inherent limitations. Recall that in our attack, the adversary records the verifier's classical query-answer pairs in a database, uses this to produce a *classical* description of a verification circuit (that does not make any oracle queries), and submits the circuit description to a PSPACE oracle in order to synthesize a money state. If the verifier instead makes quantum queries, then a natural idea is to use Zhandry's compressed oracle technique to record the quantum queries. However, there are two conceptual challenges to implementing this idea.

First, in the compresed oracle technique, the queries are being recorded by the *oracle* itself in a "database register", and not the adversary in the cryptosystem. In our setting, we are trying to construct an adversary to record the queries, but it does not have access to the oracle's database register. In general, any attempts by the adversary to get some information about the query positions of Ver could potentially disturb the intermediate states of the Ver algorithm; it is then unclear how to use the original guarantees of Ver. Another way of saying this is that Zhandry's compressed oracle technique is typically used in the *security analysis* to show limits on the adversary's ability to break some cryptosystem. In our case, we want to use some kind of quantum recording technique in the adversary's *attack*.

Secondly, the natural approach to using the PSPACE oracle is to leverage it to synthesize alleged banknotes. However, since the PSPACE oracle is a classical function (which may be accessed in superposition), it requires polynomial-length classical strings as input. In our approach, the adversary submits a classical description of a verification circuit with query/answer pairs hardcoded inside. On the other hand if Ver makes quantum queries, it may query exponentially many positions of the random oracle $\mathcal{R}$ in superposition, and it is unclear how to "squeeze" the relevant information about the queries into a polynomial-sized classical string that could be utilized by the PSPACE oracle.

This suggests that we may need a fundamentally new approach to recording quantum queries in order to handle the case when the verification algorithm makes quantum queries.

2.3 Related Work

Quantum Money. The notion of quantum money was first conceived in the paper by Wiesner [Wie83]. In the same work, a construction of private-key quantum money was proposed. Wiesner's construction has been well studied and its limitations [Lut10] and security guarantees [MVW12] have been well understood. Other constructions of private-key quantum money have also been studied. Ji, Liu and Song [JLS18] construct private-key quantum money from pseudorandom quantum states. Radian and Sattath [RS22] construct private-key quantum money with classical bank from quantum hardness of learning with errors.

With regards to public-key quantum money, Aaronson and Christiano [AC13] present a construction of public-key quantum money in the oracle model. Zhandry [Zha21] instantiated this oracle and showed how to construct public-key quantum money based on the existence of post-quantum indistinguishability obfuscation (iO) [BGI+01]. Recently, Shmueli [Shm22a] showed how to achieve public-key quantum money with classical bank, assuming post-quantum iO and quantum hardness of learning with errors. Constructions [FGH+12; KSS21; KLS22] of public-key quantum money from newer assumptions have also been explored although they have been susceptible to quantum attacks [Rob21; BDG22].

Black-box Separations in Quantum Cryptography. So far, most of the existing black-box separations in quantum cryptography have focused on extending black-box separations for classical cryptographic primitives to the quantum setting. Hosoyamada and Yamakawa [HY20] extend the black-box separation between collision-resistant hash functions and one-way functions [Sim98] to the quantum setting. Austrin, Chung, Chung, Fu, Lin and Mahmoody [ACC+22] showed a black-box separation between key agreement and one-way functions in the setting when the honest parties can perform quantum computation but only have access to classical communication. Cao and Xue [CX21] extended classical black-box separations between one-way permutations and one-way functions to the quantum setting.

3 Preliminaries

For a string x, let $|x|$ denote its length. Let $[n]$ denote the set $\{0, 1, \cdots, n-1\}$ for any positive integer n. Define the symmetric difference of two sets X and Y to be the set of elements contained in exactly one of X and Y, i.e. $X \Delta Y = (X - Y) \cup (Y - X)$.

3.1 Quantum States, Algorithms, and Oracles

A *register* R is a finite-dimensional complex Hilbert space. If $\mathsf{A}, \mathsf{B}, \mathsf{C}$ are registers, for example, then the concatenation ABC denotes the tensor product of the associated Hilbert spaces. For a linear transformation L and register R, we sometimes write L_{R} to indicate that L acts on R, and similarly we sometimes write ρ_{R} to indicate that a state ρ is in the register R. We write $\mathrm{Tr}(\cdot)$ to denote trace, and $\mathrm{Tr}_{\mathsf{R}}(\cdot)$ to denote the partial trace over a register R.

For a pure state $|\varphi\rangle$, we write φ to denote the density matrix $|\varphi\rangle\langle\varphi|$. Let I denote the identity matrix. Let $\mathrm{TD}(\rho, \sigma)$ denote the trace distance between two density matrices ρ, σ.

For a pure state $|\varphi\rangle = \sum_i a_i |i\rangle$ written in computational basis, we write $|\overline{\varphi}\rangle = \sum_i \overline{a_i} |i\rangle$ to denote the conjugate of $|\varphi\rangle$ where $\overline{a_i}$ is the complex conjugate of the complex number a_i. The following observation shows that what the maximally entangled state looks like in other basis.

Lemma 1. *For two registers* A *and* B *of the same dimension* N*, let* $(|i\rangle)_{i=1}^N$ *be the computational basis and* $(|v_i\rangle)_{i=1}^N$ *be an arbitrary basis. Then* $\frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle_{\mathsf{A}}|i\rangle_{\mathsf{B}} = \frac{1}{\sqrt{N}}\sum_{i=1}^N |v_i\rangle_{\mathsf{A}}|\overline{v_i}\rangle_{\mathsf{B}}$.

Proof. It's easy to show that $(|\overline{v_i}\rangle)_{i=1}^N$ also forms a basis. Suppose $|v_j\rangle = \sum_{i=1}^N a_{j,i}|i\rangle$. Then

$$\begin{aligned}
\frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle_{\mathsf{A}}|i\rangle_{\mathsf{B}} &= \sum_{j=1}^N |v_j\rangle\langle v_j|_{\mathsf{A}} \sum_{j'=1}^N |\overline{v_{j'}}\rangle\langle\overline{v_{j'}}|_{\mathsf{B}} \frac{1}{\sqrt{N}}\sum_{i=1}^N |i\rangle_{\mathsf{A}}|i\rangle_{\mathsf{B}} \\
&= \frac{1}{\sqrt{N}}\sum_{j=1}^N\sum_{j'=1}^N\sum_{i=1}^N \overline{a_{j,i}}a_{j',i}|v_j\rangle_{\mathsf{A}}|\overline{v_{j'}}\rangle_{\mathsf{B}} \\
&= \frac{1}{\sqrt{N}}\sum_{j=1}^N\sum_{j'=1}^N \langle v_j|v_{j'}\rangle|v_j\rangle_{\mathsf{A}}|\overline{v_{j'}}\rangle_{\mathsf{B}} \\
&= \frac{1}{\sqrt{N}}\sum_{j=1}^N |v_j\rangle_{\mathsf{A}}|\overline{v_j}\rangle_{\mathsf{B}}
\end{aligned}$$

where we use the fact that $\sum_{j=1}^N |v_j\rangle\langle v_j|_{\mathsf{A}}$ and $\sum_{j'=1}^N |\overline{v_{j'}}\rangle\langle\overline{v_{j'}}|_{\mathsf{B}}$ are identity. □

Quantum Circuits We specify the model of quantum circuits that we work with in this paper. For convenience we fix the universal gate set $\{H, CNOT, T\}$ [NC00, Chapter 4] (although our results hold for any universal gate set consisting of gates with algebraic entries). Quantum circuits can include unitary gates from the aforementioned universal gate set, as well as non-unitary gates that (a) introduce new qubits initialized in the zero state, (b) trace them out, or (c) measure them in the standard basis. We say that a circuit uses space s if the total number of qubits involved at any time step of the computation is at most s. The description of a circuit is a sequence of gates (unitary or non-unitary) along with a specification of which qubits they act on.

We call a sequence of quantum circuits $C = (C_x)_{x\in\{0,1\}^*}$ a *quantum algorithm*. We say that C is *polynomial-time* if there exists a polynomial p such that C_x has size at most $p(|x|)$. We say that C is *polynomial-space* if there exists a polynomial p such that C_x uses at most $p(|x|)$ space.

Let $C = (C_x)_{x\in\{0,1\}^*}$ denote a quantum algorithm. Given a string $x \in \{0,1\}^*$ and a state ρ whose number of qubits matches the input size of the circuit C_x, we write $C(x,\rho)$ to denote the output of circuit C_x on input ρ. The output will in general be a mixed state as the circuit C_x can perform measurements.

We say that a quantum algorithm $C = (C_x)_{x\in\{0,1\}^*}$ is *time-uniform* (or simply *uniform*) if there exists a polynomial-time Turing machine that on input x outputs the description of C_x. Similarly we say that C is *space-uniform* if there exists a polynomial-space Turing machine that on input x outputs the description of C_x.

Oracle Algorithms *Oracle algorithms* are quantum algorithms whose circuits, in addition to having the gates as described above, have the ability to query (perhaps in superposition) a function O (called an *oracle*) which may act on many qubits. This is essentially the same as the standard quantum query model [NC00, Chapter 6], except the circuits may perform non-unitary operations such as measurement, reset, and tracing out. Each oracle call is counted as a single gate towards the size complexity of a circuit. The notion of time- and space-uniformity for oracle algorithms is the

same as with non-oracle algorithms: there is a polynomial-time/polynomial-space Turing machine – which does *not* have access to the oracle – that outputs the description of the circuits.

Given an oracle $\mathcal{O} = (\mathcal{O}_n)_{n\in\mathbb{N}}$ where each $\mathcal{O}_n : \{0,1\}^n \to \{0,1\}$ is an n-bit boolean function, we write $C^{\mathcal{O}} = (C_x^{\mathcal{O}})_{x\in\{0,1\}^*}$ to denote an oracle algorithm where each circuit C_x can query any of the functions $(\mathcal{O}_n)_{n\in\mathbb{N}}$ (provided that the oracle does not act on more than the number of qubits of C_x).

In this paper we distinguish between classical and quantum queries. We say that an oracle algorithm $C^{\mathcal{O}}$ makes quantum queries if it can query $\mathcal{O}$ in superposition; this is akin to the standard query model. We say that $C^{\mathcal{O}}$ makes classical queries if, before every oracle call, the input qubits to the oracle are measured in the standard basis. In this case, the algorithm would be querying the oracle on a *probabilistic mixture* of inputs. For clarity, we write $C^{|\mathcal{O}\rangle}$ to denote C making quantum queries, and $C^{\mathcal{O}}$ to denote C making classical queries.

A specific oracle that we consider throughout is the PSPACE oracle. What we mean by this is a sequence of functions $(\mathsf{PSPACE}_n)_{n\in\mathbb{N}}$ where for every n, the function PSPACE_n decides n-bit instances of a PSPACE complete language (such as Quantified Satisfiability [Pap94]).

We state the following observation.

Lemma 2. *Let $C^{|\mathsf{PSPACE}\rangle} = (C_x^{|\mathsf{PSPACE}\rangle})_{x\in\{0,1\}^*}$ denote a polynomial-time oracle algorithm (not necessarily uniform) that makes quantum queries to PSPACE and has one-bit classical output. Then there exists a polynomial-space algorithm $D = (D_x)_{x\in\{0,1\}^*}$ such that for all $x \in \{0,1\}^*$, D_x is a unitary and the functionality of $C_x^{|\mathsf{PSPACE}\rangle}$ is exactly the same as introducing polynomial number of qubits initialized in the zero state, applying unitary D_x and then measuring the first qubit in computational basis to get a classical output. Furthermore if C is uniform, then D is space-uniform.*

Proof. This follows because for a polynomial-time (oracle) algorithm, we can always introduce new qubits only at the beginning and defer measurements and tracing out to the end, and each oracle query in $C_x^{|\mathsf{PSPACE}\rangle}$ to the PSPACE oracle can be computed by first introducing several ancillas initialized in the zero state and then applying a unitary that implements the classical polynomial space algorithm for the PSPACE-complete language and uncomputes all the intermediate results. Furthermore, the description of the unitary can be generated by polynomial-space Turing machines. □

Finally, we will consider *hybrid* oracles $\mathcal{O}$ that are composed of two separate oracles $\mathcal{R}$ and the $|\mathsf{PSPACE}\rangle$ oracle. In this model, the oracle algorithm $C^{\mathcal{O}}$ makes classical queries to $\mathcal{R}$, and quantum queries to PSPACE. We abuse the notation and refer to algorithms having access to hybrid oracles as oracle algorithms.

State Synthesis We define the following "state complexity class". Intuitively it captures the set of quantum states that can be *synthesized* by polynomial-space quantum algorithms.

Definition 1 ($\mathsf{statePSPACE}$)**.** $\mathsf{statePSPACE}$ *is the class of all sequences $(\rho_x)_{x\in S}$ for some set $S \subseteq \{0,1\}^*$ (called the* promise*) such that there is a polynomial p where each ρ_x is a density matrix on $p(|x|)$ qubits, and for every polynomial q there exists a space-uniform polynomial-space quantum algorithm $C = (C_x)_{x\in\{0,1\}^*}$ such that for all $x \in S$, the circuit C_x takes no inputs and outputs a density matrix σ such that $\mathrm{TD}(\sigma, \rho_x) \leq \exp(-q(|x|))$.*

We say that the state sequence $(\rho_x)_{x\in S}$ is pure *if each ρ_x is a pure state $|\psi_x\rangle\langle\psi_x|$; in that case we usually denote the sequence by $(|\psi_x\rangle)_{x\in S}$.*

The following theorem says that, for $\mathsf{statePSPACE}$ sequences that are pure, there in fact is a polynomial-*time* oracle algorithm that makes quantum queries to a PSPACE oracle to synthesize the state sequence.

Theorem 2 (Section 5 of [RY21])**.** *Let $(|\psi_x\rangle)_{x\in S}$ be a $\mathsf{statePSPACE}$ family of pure states. Then there exists a polynomial-time oracle algorithm $A^{|\mathsf{PSPACE}\rangle}$ such that on input $x \in S$, the algorithm outputs a pure state that is $\exp(-|x|)$-close in trace distance to $|\psi_x\rangle$.*

3.2 Public Key Quantum Money Schemes

Definition 2 (Oracle-aided Public Key Quantum Money Schemes)**.** *A* oracle-aided public key quantum money scheme $\mathcal{S}^{\mathcal{O}}$ *consists of three uniform polynomial-time oracle algorithms* $\left(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Mint}^{\mathcal{O}}, \mathsf{Ver}^{\mathcal{O}}\right)$*:*

- $\mathsf{KeyGen}^{\mathcal{O}}(1^n)$*: takes as input a security parameter n in unary notation and generates secret key-public key pair (sk, pk).*
- $\mathsf{Mint}^{\mathcal{O}}(sk)$*: takes as input sk and mints banknote ρ_s associated with the serial number s.*
- $\mathsf{Ver}^{\mathcal{O}}(pk, (s, \rho_s))$*: takes as inputs pk and an alleged banknote (s, ρ_s) and outputs $\rho'_s \otimes |\mathbf{x}\rangle\langle\mathbf{x}|$, where $\mathbf{x} \in \{\mathsf{Accept}, \mathsf{Reject}\}$.*

For simplicity, when we don't care about the output ρ'_s in $\mathsf{Ver}^{\mathcal{O}}$, we sometimes denote the event that $\rho'_s \otimes |\mathsf{Accept}\rangle\langle\mathsf{Accept}| \leftarrow \mathsf{Ver}^{\mathcal{O}}(pk, (s, \rho_s))$ as $\mathsf{Ver}^{\mathcal{O}}(pk, (s, \rho_s))$ accepts.

We require the above oracle-aided public key quantum money scheme to satisfy both correctness and security properties.

3.2.1 Correctness

We first consider the traditional definition of correctness considered by prior works. Roughly speaking, correctness states that the verification algorithm accepts the money state produced by the minting algorithm. Later, we consider a stronger notion called reusability which stipulates that the verification process on a valid money outputs another valid money state (not necessarily the same as before).

Definition 3 (Correctness)**.** *An oracle-aided public key quantum money scheme $\left(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Mint}^{\mathcal{O}}, \mathsf{Ver}^{\mathcal{O}}\right)$ is δ-**correct** if the following holds for every $n \in \mathbb{N}$:*

$$\mathsf{Pr}\left[\rho'_s \otimes |\mathsf{Accept}\rangle\langle\mathsf{Accept}| \leftarrow \mathsf{Ver}^{\mathcal{O}}(pk, (s, \rho_s)) \ : \ \begin{smallmatrix}(sk,pk)\leftarrow\mathsf{KeyGen}^{\mathcal{O}}(1^n)\\ (s,\rho_s)\leftarrow\mathsf{Mint}^{\mathcal{O}}(sk)\end{smallmatrix}\right] \geq \delta,$$

where the probability is also over the randomness of $\mathcal{O}$.

We omit δ when $\delta \geq 1 - \mathsf{negl}(n)$.

Reusability. In this work, we consider quantum money schemes satisfying the stronger notion of reusability.

Definition 4 (Reusability). *An oracle-aided public key quantum money scheme* $(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Mint}^{\mathcal{O}}, \mathsf{Ver}^{\mathcal{O}})$ *is* δ***-reusable*** *if the following holds for every* $n \in \mathbb{N}$ *and for every polynomial* $q(n)$*:*

$$\Pr\left[\rho_s^{(q(n))} \otimes |\mathsf{Accept}\rangle\langle\mathsf{Accept}| \leftarrow \mathsf{Ver}^{\mathcal{O}}(pk, (s, \rho_s^{(q(n)-1)})) \ : \ \begin{array}{c}(sk,pk)\leftarrow\mathsf{KeyGen}^{\mathcal{O}}(1^n)\\ (s,\rho_s^{(0)})\leftarrow\mathsf{Mint}^{\mathcal{O}}(sk)\\ \forall i\in[q(n)-1],\ \rho_s^{(i+1)}\otimes|\mathbf{x}\rangle\langle\mathbf{x}|\leftarrow\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(i)}))\end{array}\right] \geq \delta,$$

where the probability is also over the randomness of $\mathcal{O}$*.*

We omit δ *when* $\delta \geq 1 - \mathsf{negl}(n)$*.*

In general, gentle measurement lemma [Win99] can be invoked to prove that correctness generically implies reusability. However, this is not the case in our context. The reason being that the verification algorithm performs intermediate measurements whenever it makes classical queries to an oracle and these measurements cannot be deferred to the end.

3.2.2 Security

We consider the following security notion. Basically, it says that no efficient adversary can produce two alleged banknotes from one valid banknote with the same serial number.

Definition 5 (Security). *An oracle-aided public key quantum money scheme* $(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Mint}^{\mathcal{O}}, \mathsf{Ver}^{\mathcal{O}})$ *is* δ***-secure*** *if the following holds for every* $n \in \mathbb{N}$ *and for every uniform polynomial-time oracle algorithm* $\mathcal{A}^{\mathcal{O}}$*:*

$$\Pr\left[\mathsf{Ver}^{\mathcal{O}}(pk, (s, \phi_1)) \text{ accepts and } \mathsf{Ver}^{\mathcal{O}}(pk, (s, \phi_2)) \text{ accepts} \ : \ \begin{array}{c}(sk,pk)\leftarrow\mathsf{KeyGen}^{\mathcal{O}}(1^n)\\ (s,\rho_s)\leftarrow\mathsf{Mint}^{\mathcal{O}}(sk)\\ \phi\leftarrow\mathcal{A}^{\mathcal{O}}(pk,(s,\rho_s))\end{array}\right] \leq \delta,$$

where the probability is also over the randomness of $\mathcal{O}$*. By* ϕ_i*, we mean the reduced density matrix of* ϕ *on the* i^{th} *register.*

We omit δ *when* $\delta \leq \mathsf{negl}(n)$*.*

3.3 Jordan's Lemma and Alternating Projections

In this section, we analyze *alternating projection algorithm*, a tool for estimating the acceptance of the verification algorithm on a state with only one copy of that state, which was introduced by Marriott and Watrous [MW05] for witness-preserving error reduction. This section follows section 4.1 in [CMSZ22] mostly.

For two binary-outcome projective measurements $M_1 = \{\Pi_1, I - \Pi_1\}, M_2 = \{\Pi_2, I - \Pi_2\}$, the alternating projection algorithm applies measurements M_1 and M_2 alternatively (Π_1, Π_2 corresponds to outcome 1) until a stopping condition is met. The following lemma can help us analyze the distribution of the outcomes by decomposing it into several small subspaces.

Lemma 3 (Jordan's Lemma). *For any two projectors* Π_1*,* Π_2*, there exists an orthogonal decomposition of the Hilbert space into one-dimensional and two-dimensional subspaces* S_i *that are invariant under both* Π_1 *and* Π_2*. Moreover, if* S_i *is a one-dimensional subspace, then* Π_1 *and* Π_2 *act as identity or rank-zero projectors inside* S_i*. If* S_i *is a two-dimensional subspace, then* Π_1 *and* Π_2 *are rank-one projectors inside* S_i*. To be more specific, there are two unit vectors* $|v_i\rangle$ *and* $|w_i\rangle$ *such that inside* S_i*,* Π_1 *projects on* $|v_i\rangle$ *and* Π_2 *projects on* $|w_i\rangle$*.*

Let measurement $M_{\mathsf{Jor}} = \{P_i\}_i$ where P_i is the projector onto the subspace S_i defined above. Then both M_1 and M_2 commute with M_{Jor}. Therefore the distribution of outcomes of each M_1 and M_2 will not change if we insert M_{Jor} at any point of the alternating projections. We can analyze the distribution of outcome sequence by first applying M_{Jor} and then applying M_1, M_2 alternatively. For each two-dimensional subspace S_i, denote $p_i := |\langle v_i|w_i\rangle|^2$. (This can be seen as a quantity that measures the angle between Π_1 and Π_2 inside S_i.)

For now, let's assume that there are only two-dimensional subspaces in the decomposition. The general case where there exist one-dimensional subspaces is essentially the same and can be handled similarly. Then, $\Pi_1 = \sum_i |v_i\rangle\langle v_i|, \Pi_2 = \sum_i |w_i\rangle\langle w_i|$.[4]

Now let's state a result first proven in [MW05] and restated in many later works (e.g., [CMSZ22]).

Proposition 1. *If initially the state is $|v_i\rangle$[5] and we apply M_2, M_1 alternatively for N times, then the outcome sequence $b_1, b_2, b_3, \cdots, b_{2N}$ will follow the distribution below*

1. *Set $b_0 = 1$ (because applying M_1 to $|v_i\rangle$ will give outcome 1).*
2. *For each j, we set b_j to be b_{j-1} with probability p_i, and $1 - b_{j-1}$ otherwise.*

Moreover, whenever we measure M_1 and get outcome 1, we will go back to state $|v_i\rangle$.

Then the fraction of bit-flips in the outcome sequence will be a good estimation of $1 - p_i$ if we start from $|v_i\rangle$.

3.4 Compressed Oracle Techniques

In this section, we present some basics of compressed oracle techniques introduced by Zhandry [Zha18].

For a quantum query algorithm A interacting with a random oracle, let's assume that A only queries the random oracle with n-bit input and gets 1-bit output for simplicity. By the deferred measurement principle, without loss of generality we can write A in the form of a sequence of unitaries $U_0, U_f, U_1, U_f, \cdots, U_{k-1}, U_f, U_k$ where U_i is the unitary that prepares the $(i+1)^{th}$ query of A to R and U_f maps $|x\rangle|y\rangle$ to $|x\rangle|y \oplus f(x)\rangle$ where f is the chosen random function from all the functions with n-bit input and 1-bit output.

Then the behavior of A when it is interacting with a random oracle can be analyzed in the following *purified view*:

- Initialize register A to be the input for A (along with enough ancillas $|0\rangle$) and initialize register F to be a uniform superposition of the truth tables of all functions from $[2^n]$ to $\{0,1\}$ (to be more specific, F is initialized to $\frac{1}{\sqrt{2^{2^n}}} \sum_{f \text{ is a } 2^n\text{-bit string}} |f\rangle$ where $|f\rangle = |f(0)\rangle|f(1)\rangle\cdots|f(2^n-1)\rangle$ and $|f(i)\rangle$ consists of one qubit).
- Apply $U_0, U_F, U_1, U_F, \cdots, U_{k-1}, U_F, U_k$ where U_i is acting on A and U_F maps $|x\rangle|y\rangle|f\rangle_{\mathsf{F}}$ to $|x\rangle|y \oplus f(x)\rangle|f\rangle_{\mathsf{F}}$.

[4] Generally, for each one-dimensional subspace S_i on which Π_1 acts as identity, we can set $|v_i\rangle$ to be the vector that spaces S_i. Let A be the set of index i such that Π_1 is not a rank-zero projector inside S_i. Then $\Pi_1 = \sum_{i\in A} |v_i\rangle\langle v_i|$. Similarly $\Pi_2 = \sum_{i\in B} |w_i\rangle\langle w_i|$ where B and $|w_i\rangle$ are defined in a similar way.

[5] The same holds for each $|v_i\rangle$ $(i \in A)$ generally where we define $p_i = 1$ if Π_1 and Π_2 act both as identity in subspace S_i and we define $p_i = 0$ if Π_1 acts as identity while Π_2 acts as zero-projector in subspace S_i.

In fact, the output (mixed) state of A (we also take the randomness of f into account) equals to the reduced density matrix on the output register of the state we obtain from the above procedure as U_i, U_F commutes with computational basis measurement on F. More generally, the output (mixed) state of a sequence of algorithms with access to random oracle can also be analyzed in this way.

Definition 6 (Fourier basis)**.** $|\hat{0}\rangle := |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. $|\hat{1}\rangle := |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

One can easily check that $\{|\hat{0}\rangle, |\hat{1}\rangle\}$ *is a basis because it's just the result of applying hermitian matrix* H *to* $|0\rangle, |1\rangle$*. We call this basis as* Fourier basis*.*

The following fact is simple and easy to check, but crucial in compressed oracle techniques. Roughly speaking, it says that if we see $CNOT$ in Fourier basis, its control bit and target bit swaps.

Fact 1. *The operator defined by* $|y\rangle|y'\rangle \to |y \oplus y'\rangle|y'\rangle$ *for all* $y, y' \in \{0,1\}$ *is the same as the operator defined by* $|\widehat{y}\rangle|\widehat{y'}\rangle \to |\widehat{y}\rangle|\widehat{y' \oplus y}\rangle$ *for all* $y, y' \in \{0,1\}$*.*

By Fact 1, when we look at the last two registers in Fourier basis, U_F becomes

$$|x\rangle|\widehat{y}\rangle|\widehat{y_0}\rangle|\widehat{y_1}\rangle\cdots|\widehat{y_{2^n-1}}\rangle \to |x\rangle|\widehat{y}\rangle|\widehat{y_0}\rangle|\widehat{y_1}\rangle\cdots|\widehat{y_{x-1}}\rangle|\widehat{y_x \oplus y}\rangle|\widehat{y_{x+1}}\rangle\cdots|\widehat{y_{2^n-1}}\rangle.$$

Initially, F is $|\hat{0}\rangle|\hat{0}\rangle\cdots|\hat{0}\rangle$ and each call of U_F only changes one position if we look at the last two registers in Fourier basis. So after k calls of U_F, the state can be written as

$$\sum_{\substack{a,y_0,y_1,\cdots,y_{2^n-1} \\ \text{such that there are at most } k \text{ non-zero} \\ \text{in } y_0, y_1, \cdots, y_{2^n-1}}} \alpha_{a,y_0,y_1,\cdots,y_{2^n-1}} |a\rangle_{\mathsf{A}} |\widehat{y_0}\rangle|\widehat{y_1}\rangle\cdots|\widehat{y_{2^n-1}}\rangle.$$

We can record those non-$\hat{0}$ into a database. To be more specific, there exists a unitary that maps those $|\widehat{y_0}\rangle|\widehat{y_1}\rangle\cdots|\widehat{y_{2^n-1}}\rangle$ (perhaps along with some ancillas) to a database $|x_1\rangle|\widehat{y_{x_1}}\rangle\cdots|x_l\rangle|\widehat{y_{x_l}}\rangle$ (perhaps along with some unused space) where $x_1 < x_2 < \cdots < x_l, \widehat{y_{x_i}} \neq \hat{0}$ and $l \leq k$. That is, there exists a unitary that can compress the oracle. Furthermore, the inverse of the unitary can decompress the database back to the oracle.

Chernoff bound Finally, we state here a variant of the Chernoff bound that we will use.

Theorem 3 (Chernoff Bound)**.** *Suppose* $X_1, X_2, \cdots, X_n$ *are independent random variables taking values from* $\{0,1\}$ *such that each* $X_i = 1$ *with probability* p*. Let* $\mu = pn$*. Then for any* $\delta > 0$*,*

$$\mathsf{Pr}\left[\sum_{i=1}^{n} X_i \geq (1+\delta)\mu\right] \leq e^{-\delta^2\mu/(2+\delta)},$$

$$\mathsf{Pr}\left[\sum_{i=1}^{n} X_i \leq (1-\delta)\mu\right] \leq e^{-\delta^2\mu/2}.$$

4 Synthesizing Witness States In Quantum Polynomial Space

In the classical setting it is easy to see that given a (classical) verifier circuit V (which may make oracle queries to PSPACE), one can find in polynomial space a witness string y that is accepted by V: one can simply perform brute-force search over all strings and check whether V^{PSPACE} accepts x.

In the section, we prove the quantum counterpart, where now the verifier circuit is quantum and can make quantum queries to the PSPACE oracle. We show that given the description of such a verifier circuit, with the help of the quantum $|\mathsf{PSPACE}\rangle$ oracle, we can efficiently *synthesize* a witness state ρ that is accepted by V with probability greater than the desired guarantee (provided that there exists a witness state with acceptance probability greater than the threshold). Formally:

Theorem 4. *Let a (called the* guarantee*), b (called the* threshold*) be functions such that $b(n) - a(n) \geq \frac{1}{p(n)}$ for every n where p is a polynomial. Let $\mathsf{V}^{|\mathsf{PSPACE}\rangle}$ denote a uniform oracle algorithm. Then there exists a uniform oracle algorithm Syn (called the* synthesizer*) such that for every $x \in S$,*

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \mathsf{Syn}^{|\mathsf{PSPACE}\rangle}(x)) \text{ accepts}\right] \geq a(|x|)$$

where $S := \left\{x : \max_\rho \mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \rho) \text{ accepts}\right] \geq b(|x|)\right\}$.

This theorem follows directly from Theorem 2 and the following lemma.

Lemma 4. *Let a, b be functions such that $b(n) - a(n) \geq \frac{1}{p(n)}$ for every n where p is a polynomial. Let $\mathsf{V}^{|\mathsf{PSPACE}\rangle}$ denote a uniform oracle algorithm, and let S be the corresponding set as in Theorem 4. Then there exists a $\mathsf{statePSPACE}$ family of pure states $(|\psi_x\rangle)_{x\in S}$ where each state $|\psi_x\rangle$ is bipartite on two registers (labeled M and E) such that for every $x \in S$,*

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \rho_{\mathsf{M}}) \text{ accepts}\right] \geq a(|x|)$$

where ρ_{M} is the reduced density matrix of $|\psi_x\rangle$ on register M, i.e. $\rho_{\mathsf{M}} = \mathrm{Tr}_{\mathsf{E}}(\psi_x)$.

Proof of Theorem 4. Let $a'(n) = a(n) + \exp(-n)$ and $b'(n) = b(n)$, where $a(n), b(n)$ are as given by the conditions in Theorem 4. Applying Lemma 4 with functions $a'(n), b'(n)$, we obtain a $\mathsf{statePSPACE}$ state sequence $(|\psi_x\rangle)_{x\in S}$ such that for every $x \in S$,

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \rho_{\mathsf{M}}) \text{ accepts}\right] \geq a(|x|) + \exp(-|x|)$$

where $S := \left\{x : \max_\rho \mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \rho) \text{ accepts}\right] \geq b(|x|)\right\}$.

Theorem 2 implies that there exists a polynomial-time oracle algorithm $A^{|\mathsf{PSPACE}\rangle}$ that on input $x \in S$, outputs a pure state $|\varphi_x\rangle$ that is $\exp(-|x|)$-close to $|\psi_x\rangle$. This implies that the reduced density matrix of $A^{|\mathsf{PSPACE}\rangle}(x)$ on register M, which we denote by σ_{M}, is also $\exp(-|x|)$-close to $\rho_{\mathsf{M}} = \mathrm{Tr}_{\mathsf{E}}(\psi_x)$ (this follows from the fact that trace distance is non-increasing when you discard subsystems). Thus for every $x \in S$, we have

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \sigma_{\mathsf{M}}) \text{ accepts}\right] \geq a(|x|)$$

because otherwise $\mathsf{V}^{|\mathsf{PSPACE}\rangle}$ would be able to distinguish between ρ_{M} and σ_{M} with more than $\exp(-|x|)$ bias.

The synthesizer Syn works as follows: on input $x \in S$ it runs the oracle algorithm $A^{|\mathsf{PSPACE}\rangle}(x)$ to obtain a pure state $|\varphi_x\rangle$, and then traces out the E register and returns the remaining state on the M register as output.

□

The remainder of Section 4 will be devoted to the proof of Lemma 4. We will use the techniques and results from [MW05] (also presented in Section 3.3 for completeness). In Section 4.1 we present the description of the state family along with the description of a circuit family that generates (an approximation of) the state family. In Section 4.2 we prove that the state family satisfies the requirements.

4.1 Description of the State Family and Circuit Family

In this section, we implement our ideas from Section 2.1 in a formal way. Recall that our algorithm in Section 2.1 repeatedly does the following (which we will call a *trial*): start from a maximally entangled state, estimate the acceptance probability coherently using MW technique and if the estimated acceptance probability high, then output the remaining state. Roughly speaking, the target state we aim to generate will be the remaining state after a successful trial (a trial is successful if the estimated acceptance probability is high). Looking ahead, in order to prove Lemma 4, we only need to show two things. Firstly, our algorithm actually outputs a good approximation of the target state, so our target state forms a $\mathsf{statePSPACE}$ family; Secondly, our target state will indeed be accepted with high probability.

Now let's start by giving a formal description of the state family.

The state family Let $\mathsf{V}^{|\mathsf{PSPACE}\rangle}$ be the uniform oracle algorithm given in the condition of Lemma 4. From Lemma 2, there exists a space-uniform polynomial-space algorithm $\widehat{\mathsf{V}} = (\widehat{\mathsf{V}}_x)_{x\in\{0,1\}^*}$ such that $\widehat{\mathsf{V}}_x$ is unitary and the functionality of $\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}$ is exactly the same as introducing $k(|x|)$ new ancilla qubits in $|0\rangle$, applying unitary $\widehat{\mathsf{V}}_x$ and then measuring the first qubit in computational basis where k is a polynomial. Let $m(|x|)$ be the number of qubits that V_x takes as input, which is also a polynomial.

Fix $x \in S, n = |x|$. We sometimes omit subscript x when it is clear from the context. For convenience, we write $m(n), k(n)$ as m, k respectively from now on.

Let M denote the register containing the m input qubits. Let K denote the register containing the k ancilla qubits. Let Ans denote the first qubit (i.e. the one that will be measured in computational basis to decide whether $\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}$ accepts or rejects, and outcome 1 means accept while outcome 0 means reject). Let Aux denote a register containing another m fresh qubits.

Here we define two binary-outcome projective measurements on MK. Define $P^1 := |0^k\rangle\langle 0^k|_{\mathsf{K}}$, $P^0 := I_{\mathsf{MK}} - P^1$ and $P := \{P^0, P^1\}$. Intuitively, P^1 corresponds to "valid input subspace" (i.e., the ancilla qubits are initialized properly). Define $Q^1 := \widehat{\mathsf{V}}_x^\dagger |1\rangle\langle 1|_{\mathsf{Ans}} \widehat{\mathsf{V}}_x, Q^0 := \widehat{\mathsf{V}}_x^\dagger |0\rangle\langle 0|_{\mathsf{Ans}} \widehat{\mathsf{V}}_x$ and $Q := \{Q^0, Q^1\}$. Intuitively, Q^1 corresponds to the state that will be accepted if we apply $\widehat{\mathsf{V}}_x$ to it and then measure Ans in computational basis. So Q checks whether $\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}$ will accept as long as register K is initialized properly. The following simple observation is implicitly shown in [MW05].

Observation 1. *The maximum acceptance probability of* $\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x, \cdot)$ *is exactly the largest eigenvalue of* $P^1 Q^1 P^1$.

Proof. First, we show that the maximum eigenvalue of $P^1Q^1P^1$ is upper bounded by the maximum acceptance probability of $\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\cdot)$.

For any pure state $|\phi\rangle$ on register MK, let $\rho_{\mathsf{M}} = \frac{1}{\|P^1|\phi\rangle\|^2}\mathrm{Tr}_{\mathsf{K}}(P^1|\phi\rangle\langle\phi|P^1)$. Then

$$\begin{aligned}\langle\phi|P^1Q^1P^1|\phi\rangle =& \mathrm{Tr}(Q^1P^1|\phi\rangle\langle\phi|P^1) = \|P^1|\phi\rangle\|^2\mathrm{Tr}(Q^1\frac{1}{\|P^1|\phi\rangle\|^2}P^1|\phi\rangle\langle\phi|P^1)\\ =&\|P^1|\phi\rangle\|^2\mathrm{Tr}(Q^1(\rho_{\mathsf{M}}\otimes|0^k\rangle\langle 0^k|_{\mathsf{K}}))\\ =&\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\rho_{\mathsf{M}})\text{ accepts}\right]\\ \leq&\max_{\rho}\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\rho)\text{ accepts}\right]\end{aligned}$$

Second, we show that the maximum eigenvalue of $P^1Q^1P^1$ is lower bounded by the maximum acceptance probability of $\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\cdot)$. By a simple convexity argument, we can assume without loss of generality, the acceptance probability of $\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\cdot)$ achieves its maximum on pure state $|\phi_0\rangle$. Let $|\phi\rangle = |\phi_0\rangle|0^k\rangle$. Then

$$\langle\phi|P^1Q^1P^1|\phi\rangle = \mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\phi_0)\text{ accepts}\right] = \max_{\rho}\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\rho)\text{ accepts}\right]$$

Therefore, this observation holds true.

□

We first define a subroutine Trial where $N := \max\left(\frac{3a+b}{(b-a)^2}\left(m+2-\log(b-a)\right), \frac{16b}{(b-a)^2}\right)$ is polynomial in n (recall that $b(n)-a(n)\geq\frac{1}{p(n)}$ where p is a polynomial).

1: Initialize register MAux to be $\frac{1}{\sqrt{2^m}}\sum_{i\in\{0,1\}^m}|i\rangle_{\mathsf{M}}|i\rangle_{\mathsf{Aux}}$
2: Initialize register K to be $|0^k\rangle_{\mathsf{K}}$
3: Introduce a new $2N+1$ qubit register $\mathsf{Y} := \mathsf{Y_0}\cdots\mathsf{Y_{2N}}$ initialized to be $|1\rangle|0^{2N}\rangle$
4: Introduce a new register Cnt initialized in $|0\rangle$
5: **for** $i = 1, 2, \cdots, N$ **do**
6: Measure MK with Q coherently, store the outcome in Y_{2i-1}
7: Measure MK with P coherently, store the outcome in Y_{2i}
8: **end for**
9: Compute the number of times that $y_j = y_{j-1}$ in superposition and store the result in Cnt
10: Do the projective measurement $\mathsf{Test} := \{\mathsf{Yes} := \sum_{j\geq N(a+b)}|j\rangle\langle j|_{\mathsf{Cnt}}\otimes|1\rangle\langle 1|_{\mathsf{Y}_{2N}}, \mathsf{No} := I-\mathsf{Yes}\}$

Define $\mathsf{E} := \mathsf{K}\otimes\mathsf{Aux}\otimes\mathsf{Y}\otimes\mathsf{Cnt}$ (i.e., all registers except M).

Definition 7 (State family $(|\psi_x\rangle)_{x\in S}$)**.** *Let S be the set defined in Lemma 4. When $x\in S$, let $|\psi_x\rangle$ denote the state in register $\mathsf{M}\otimes\mathsf{E}$ after a successful implementation of* Trial *(i.e., the outcome of* Test *is* Yes*). When x are clear from the context, we also write it as $|\psi\rangle$.*

Observe that in Trial, we initialize a pure state in register $\mathsf{M}\otimes\mathsf{E}$ (line 1 - line 4), then apply a unitary on it (line 5- line 9) as all measurements are conducted coherently, and finally do a projective measurement (line 10). So the definition above indeed gives us a family of states $(|\psi_x\rangle)_{x\in S}$ such that each $|\psi_x\rangle$ is a pure state on $l(n)$ qubits where l is a polynomial.

The circuit family Now let's construct a circuit family (or algorithm) that can generate efficiently an approximation of the state family $(|\psi_x\rangle)_{x\in S}$. For any polynomial q and approximation factor $\exp(-q(n))$, the circuit C_x operates as follows where $T := 2^{m+2}q(n)$ is exponential in n.

1: **for** $t = 1, \cdots, T$ **do**
2: Run Trial
3: **if** it is successful **then**
4: **return** the state in the register ME
5: **end if**
6: **end for**
7: **return** an arbitrary state with $l(n)$ qubits

4.2 Proof of Lemma 4

In the section, we prove that the pure state family $(|\psi_x\rangle)_{x\in S}$ satisfies the requirements in Lemma 4 by applying known result in Section 3.3.

Fix $x \in S$. We associate Π_1 with P^1, Π_2 with Q^1, M_1 with P and M_2 with Q, and adopt the notations in Section 3.3. Then $P^1Q^1P^1 = \sum_i |v_i\rangle\langle v_i| \sum_i |w_i\rangle\langle w_i| \sum_i |v_i\rangle\langle v_i| = \sum_i p_i |v_i\rangle\langle v_i|$. From $x \in S$ and Observation 1, we can assume $p_1 = \max_i p_i \geq b$. [6]

To begin with, let's prove that the state family $(|\psi_x\rangle)_{x\in S}$ satisfies the first requirement in Lemma 4. That is, it is a statePSPACE family, which can be approximately generated by the circuit family. Notice that C_x outputs $|\psi_x\rangle$ whenever one of the exponential Trials succeeds. So we first analyze the success probability of one Trial.

Lemma 5. $\Pr[\text{Trial } \textit{succeeds}] \geq \frac{1}{2^{m+2}}$.

Proof. The success probability of Trial doesn't change if we measure each qubit of Y once the outcome is stored in it because computational basis measurements on Y commutes with the operations in line 9 and 10 of Trial. Thus we can think of it as measure MK directly with P and Q alternatively, get a classical outcome sequence $y := y_1 y_2 \cdots y_{2N}$ and return Yes if $y_{2N} = 1$ and the number of times that $y_j = y_{j-1} (1 \leq j \leq 2N)$ is at least $N(a + b)$ (where $y_0 = 1$), which is an alternating projection algorithm. For simplicity, we will denote by Good the set of y that corresponds to outcome Yes. Now let's analyze the probability of $y \in$ Good.

An important observation is that the initial state can also be written in forms of $|v_i\rangle$. Because $P^1 = \sum_i |v_i\rangle\langle v_i|$, $|v_i\rangle$ forms a basis for the Hilbert space $\mathcal{H}_{\mathsf{M}} \otimes |0^k\rangle\langle 0^k|_{\mathsf{K}}$. Let $|u_i\rangle$ be a truncation on M of $|v_i\rangle$, i.e. $|v_i\rangle_{\mathsf{MK}} = |u_i\rangle_{\mathsf{M}} |0^k\rangle_{\mathsf{K}}$. Then $|u_i\rangle$ forms a basis for $\mathcal{H}_{\mathsf{M}}$. Thus by Lemma 1, the state

$$\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle_{\mathsf{M}} |i\rangle_{\mathsf{Aux}} = \frac{1}{\sqrt{2^m}} \sum_i |u_i\rangle_{\mathsf{M}} |\overline{u_i}\rangle_{\mathsf{Aux}}$$

Consequently, $\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle_{\mathsf{M}} |0^k\rangle_{\mathsf{K}} |i\rangle_{\mathsf{Aux}} = \frac{1}{\sqrt{2^m}} \sum_i |v_i\rangle_{\mathsf{MK}} |\overline{u_i}\rangle_{\mathsf{Aux}}$. [7]

Notice that we can apply M_{Jor} on register MK before the alternating projections without changing the distribution of y. Applying M_{Jor} to the above state, the post-measurement state will

[6] Generally $P^1Q^1P^1 = \sum_{i\in A\cap B} p_i |v_i\rangle\langle v_i|$. We can assume $p_1 = \max_{i\in A\cap B} p_i \geq b$.

[7] In the general case, the summation is over $i \in A$. That is, $\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle_{\mathsf{M}} |0^k\rangle_{\mathsf{K}} |i\rangle_{\mathsf{Aux}} = \frac{1}{\sqrt{2^m}} \sum_{i\in A} |v_i\rangle_{\mathsf{MK}} |\overline{u_i}\rangle_{\mathsf{Aux}}$. Thus for each $i \in A$, we will be in subspace S_i with probability $\frac{1}{2^m}$ if we apply M_{Jor}.

be $|v_i\rangle_{\mathsf{MK}}|\overline{u_i}\rangle_{\mathsf{Aux}}$ with probability $\frac{1}{2^m}$. And by Proposition 1, when we start from $|v_i\rangle_{\mathsf{MK}}|\overline{u_i}\rangle_{\mathsf{Aux}}$, $y_j = y_{j-1}$ with probability p_i for each j independently.

In particular, we will start from $|v_1\rangle_{\mathsf{MK}}|\overline{u_1}\rangle_{\mathsf{Aux}}$ with probability $\frac{1}{2^m}$. And when we start from $|v_1\rangle_{\mathsf{MK}}|\overline{u_1}\rangle_{\mathsf{Aux}}$, $y_j = y_{j-1}$ with probability p_1 for each j independently. This can be seen as performing $2N$ independent coin flips with bias $p_1 \geq b$. And $y \in \mathsf{Good}$ if the number of heads (denote as cnt) is an even greater than or equal to $N(a+b)$.

By Chernoff bound,

$$\Pr[cnt < (a+b)N] \leq \exp(-Np_1(1-\frac{a+b}{2p_1})^2) \leq \exp(-N\frac{(b-a)^2}{4b}) \leq \frac{1}{4}$$

$$\begin{aligned}\Pr[cnt \text{ is an odd}] =& \sum_{j=0}^{N-1}\binom{2N}{2j+1}p_1^{2j+1}(1-p_1)^{2N-2j-1}\\ =&\frac{1}{2}((p_1+1-p_1)^{2N}-(p_1-(1-p_1))^{2N})\\ \leq&\frac{1}{2}\end{aligned}$$

Thus by union bound, when the post-measurement state after M_{Jor} is $|v_1\rangle_{\mathsf{MK}}|\overline{u_1}\rangle_{\mathsf{Aux}}$, $y \in \mathsf{Good}$ with probability at least $\frac{1}{4}$.

So $\Pr[\mathsf{Trial} \text{ succeeds}] = \Pr[y \in \mathsf{Good}] \geq \frac{1}{2^m}\frac{1}{4} = \frac{1}{2^{m+2}}$.

□

Claim 1. *$(|\psi_x\rangle)_{x\in S}$ is a $\mathsf{statePSPACE}[S]$ family.*

Proof. We only need to show that our construction C_x satisfies the requirements in Definition 1.

From the construction, C_x can be generated by polynomial space Turing machine on input x and C_x uses at most polynomial space at any time. Thus $C = (C_x)_{x\in\{0,1\}^*}$ is a space-uniform polynomial-space quantum algorithm. It is obvious from the construction that C_x takes no inputs. The only remaining thing is to prove C_x outputs a good approximation of $|\psi_x\rangle$ when $x \in S$.

Fix $x \in S$. Whenever there is a successful implementation of Trial, C_x will output $|\psi_x\rangle$. Moreover, by Lemma 5, recall that $T = 2^{m+2}q(n)$,

$$\Pr[T \text{ independent repetitions of } \mathsf{Trial} \text{ all fail}] = (1-\Pr[\mathsf{Trial} \text{ succeeds}])^T \leq (1-\frac{1}{2^{m+2}})^T \leq e^{-q(n)}$$

That is, except with probability $e^{-q(n)}$, C_x outputs $|\psi_x\rangle$.

As a result, the state outputted by C_x is $e^{-q(n)}$-close to $|\psi_x\rangle$ in trace distance. □

The second requirement in Lemma 4 that $(|\psi_x\rangle)_{x\in S}$ needs to satisfy is that the reduced density matrix of $|\psi_x\rangle$ will be accepted by $\mathsf{V}^{|\mathsf{PSPACE}\rangle}(x,\cdot)$ with high probability. This is intuitively correct because the real acceptance probability should not be too far from the estimated acceptance probability. Now let's prove it formally.

Claim 2. *For every $x \in S$, $\Pr\left[\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}(\rho_{\mathsf{M}}) \text{ accepts}\right] \geq a$ where ρ_{M} is the reduced density matrix of ψ_x on register M.*

Proof. Fix $x \in S$. We will omit subscripts when it is clear from the context.

Similar with what we did in Lemma 5, this probability doesn't change if in the generation of ψ_x (i.e. Trial), we measure Y directly instead (as we only care about the part in M). Let ψ'_x be the state we obtain from a successful implementation of Trial if we measure directly instead. Then the reduced density matrices of ψ_x and ψ'_x are the same on register M.

Notice that $\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}(\rho_{\mathsf{M}})$ is just applying $\widehat{\mathsf{V}}_x$ to $\rho_{\mathsf{M}} \otimes |0^k\rangle\langle 0^k|_{\mathsf{K}}$ and then measuring the first qubit in computational basis, or equivalently, it is just measuring $\rho_{\mathsf{M}} \otimes |0^k\rangle\langle 0^k|_{\mathsf{K}}$ with Q. By the definition, ψ'_x is $|0^k\rangle$ on register K (because the outcome y_{2N} should be 1). So the reduced density matrix of ψ'_x on register MK is exactly $\rho_{\mathsf{M}} \otimes |0^k\rangle\langle 0^k|_{\mathsf{K}}$.

Consider the following alternating projection algorithm:

We start from $\frac{1}{\sqrt{2^m}} \sum_{i\in\{0,1\}^m} |i\rangle_{\mathsf{M}} |0^k\rangle_{\mathsf{K}} |i\rangle_{\mathsf{Aux}}$, apply Q, P to the state alternatively for N times to obtain a classical outcome sequence $y := y_1 y_2 \cdots y_{2N}$ and if y meets the requirement (to be more accurate, $y \in \mathsf{Good}$ where Good is defined in Lemma 5), we will additionally apply Q to get an outcome z and accept if $z = 1$.

In the above algorithm, if $y \in \mathsf{Good}$, the (mixed) state remaining is exactly ψ'_x, whose reduced density matrix on MK is $\rho_{\mathsf{M}} \otimes |0^k\rangle\langle 0^k|_{\mathsf{K}}$. Recall that $\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}(\rho_{\mathsf{M}})$ is just measuring $\rho_{\mathsf{M}} \otimes |0^k\rangle\langle 0^k|_{\mathsf{K}}$ with Q. Therefore,

$$\mathsf{Pr}\left[\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}(\rho_{\mathsf{M}}) \text{ accepts}\right] = \mathsf{Pr}\left[z = 1 \mid y \in \mathsf{Good}\right]$$

From Section 3.3, $\mathsf{Pr}\left[z = 1 \mid y \in \mathsf{Good}\right]$ will not change if we insert M_{Jor} in front of the alternating projections. So we can calculate it by projecting the initial state $\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle_{\mathsf{M}} |0^k\rangle_{\mathsf{K}} |i\rangle_{\mathsf{Aux}}$ to one of the subspaces S_i, getting the post-measurement state $|v_i\rangle_{\mathsf{MK}} |\overline{u_i}\rangle_{\mathsf{Aux}}$ and then sampling y and z as if we start from this state (here we also use the fact that $\frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle_{\mathsf{M}} |0^k\rangle_{\mathsf{K}} |i\rangle_{\mathsf{Aux}}$ can be written in the form $\frac{1}{\sqrt{2^m}} \sum_i |v_i\rangle_{\mathsf{MK}} |\overline{u_i}\rangle_{\mathsf{Aux}}$). Denote E_i be the event that we get the post-measurement state $|v_i\rangle_{\mathsf{MK}} |\overline{u_i}\rangle_{\mathsf{Aux}}$. Then $\mathsf{Pr}\left[E_i\right] = \frac{1}{2^m}$[8]. Therefore,

$$\begin{aligned}
\mathsf{Pr}\left[z = 1 \mid y \in \mathsf{Good}\right] =& \frac{\mathsf{Pr}\left[z = 1 \wedge y \in \mathsf{Good}\right]}{\mathsf{Pr}\left[y \in \mathsf{Good}\right]} \\
=& \frac{\sum_i \mathsf{Pr}\left[E_i\right] \mathsf{Pr}\left[y \in \mathsf{Good} \mid E_i\right] \mathsf{Pr}\left[z = 1 \mid E_i \wedge y \in \mathsf{Good}\right]}{\sum_i \mathsf{Pr}\left[E_i\right] \mathsf{Pr}\left[y \in \mathsf{Good} \mid E_i\right]} \\
=& \frac{\sum_i p_i \mathsf{Pr}\left[y \in \mathsf{Good} \mid E_i\right]}{\sum_i \mathsf{Pr}\left[y \in \mathsf{Good} \mid E_i\right]}
\end{aligned}$$

Same as Lemma 5, when we start from $|v_i\rangle_{\mathsf{MK}} |\overline{u_i}\rangle_{\mathsf{Aux}}$, the probability of $y \in \mathsf{Good}$ is the same as the probability that during $2N$ independent coin flips with bias p_i, the number of heads (denote as cnt) is an even greater than or equal to $N(a + b)$.

So if $p_i < a$, by Chernoff bound,

$$\mathsf{Pr}\left[y \in \mathsf{Good} \mid E_i\right] \leq \mathsf{Pr}\left[cnt \geq N(a+b)\right] \leq \exp(-2Np_i(\frac{a+b}{2p_i} - 1)^2/(1 + \frac{a+b}{2p_i})) \leq \exp(-N\frac{(b-a)^2}{3a+b})$$

[8] Generally, $\mathsf{Pr}\left[E_i\right] = \frac{1}{2^m}$ for each $i \in A$. And thus all the summations below will be only over $i \in A$.

From Lemma 5, $\mathsf{Pr}[y \in \mathsf{Good} \mid E_1] \geq \frac{1}{4}$. As a result,

$$\sum_{p_i \geq a}(p_i - a)\mathsf{Pr}[y \in \mathsf{Good} \mid E_i] - \sum_{p_i < a}(a - p_i)\mathsf{Pr}[y \in \mathsf{Good} \mid E_i] \geq (b-a)\frac{1}{4} - 2^m a \exp(-N\frac{(b-a)^2}{3a+b}) > 0$$

where we use the fact that there are only 2^m $|v_i\rangle$s because P^1 has rank 2^m.[9]

The above inequality can be rearranged into $\sum_i p_i \mathsf{Pr}[y \in \mathsf{Good} \mid E_i] > a \sum_i \mathsf{Pr}[y \in \mathsf{Good} \mid E_i]$. Therefore, $\mathsf{Pr}\left[\mathsf{V}_x^{|\mathsf{PSPACE}\rangle}(\rho_{\mathsf{M}}) \text{ accepts}\right] \geq a$, which ends the proof of this claim. □

Lemma 4 follows directly from the above two claims.

5 Insecurity of Oracle-Aided Public-Key Quantum Money

In this section, we will use the synthesizer from Section 4 as a building block to attack the oracle-aided public key quantum money scheme where $\mathcal{O}$ is a hybrid oracle composed of random oracle $\mathcal{R}$ and $|\mathsf{PSPACE}\rangle$. Formally:

Theorem 5. *Reusable and secure oracle-aided public key quantum money scheme* $(\mathsf{KeyGen}^{\mathcal{R},|\mathsf{PSPACE}\rangle}, \mathsf{Mint}^{\mathcal{R},|\mathsf{PSPACE}\rangle}, \mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle})$ *does not exist where* $\mathcal{R}$ *is a random oracle.*

Informally speaking, our synthesizer in Theorem 4 works for uniform oracle algorithm $\mathsf{V}^{|\mathsf{PSPACE}\rangle}$. However, in the oracle-aided public key quantum money scheme we aim to attack, the verification algorithm has access to random oracle $\mathcal{R}$ in addition to $|\mathsf{PSPACE}\rangle$. Inspired by [CKP15; AK22], we try to remove $\mathcal{R}$ and simulate it with a good database. Based on the ideas in Section 2.2, we give the following attacker.

Let $\mathcal{O}$ be the hybrid oracle composed of random oracle $\mathcal{R}$ and $|\mathsf{PSPACE}\rangle$. For a δ_r-reusable δ_s-secure oracle-aided quantum money scheme $(\mathsf{KeyGen}^{\mathcal{O}}, \mathsf{Mint}^{\mathcal{O}}, \mathsf{Ver}^{\mathcal{O}})$ where $\delta_r = 0.99, \delta_s = \mathsf{negl}(n)$, denote $l(n)$ to be the number of queries to $\mathcal{R}$ made by one execution of $\mathsf{KeyGen}^{\mathcal{O}}$ and one execution of $\mathsf{Mint}^{\mathcal{O}}$. By efficiency of $\mathsf{Ver}^{\mathcal{O}}$, there exists a uniform oracle algorithm $\mathsf{V}^{|\mathsf{PSPACE}\rangle} = (\mathsf{V}_x^{|\mathsf{PSPACE}\rangle})_{x\in\{0,1\}^*}$ such that running $\mathsf{V}_{(pk,D,s)}^{|\mathsf{PSPACE}\rangle}(\rho)$ is the same as running $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho))$ where $\mathcal{R}$ is simulated with database D.

Let $\epsilon = 0.01$, $b = 1 - \sqrt{1-\delta_r+\epsilon}$, $a = 0.99b$. By Theorem 4, there exists a polynomial-time uniform oracle algorithm $\mathsf{Syn}^{|\mathsf{PSPACE}\rangle}$ which can generate an "almost optimal" witness state of $\mathsf{V}_{(pk,D,s)}^{|\mathsf{PSPACE}\rangle}$ with guarantee a and threshold b. Now let's construct the adversary $\mathcal{A}^{\mathcal{O}}$.

Adversary $\mathcal{A}^{\mathcal{O}}$. It takes as input a valid banknote (s,ρ_s) and public key pk, and behaves as follows.

1. Let $t \xleftarrow{\$} \{0,\ldots,\lceil\frac{\ell}{\epsilon}\rceil - 1\}$. Let $D = \emptyset$, $\rho_s^{(0)} = \rho_s$. Run the following t times. In i^{th} iteration,
   (a) $\rho_s^{(i)} \otimes |\mathbf{x}\rangle\langle\mathbf{x}| \leftarrow \mathsf{Ver}^{\mathcal{O}}(pk,s,\rho_s^{(i-1)})$.
   (b) Add query-answer pairs to $\mathcal{R}$ in item (a) into D.

[9] For the general case, we only sum over $i \in A$ and $|A|$ equals to the rank of P^1, which is 2^m.

2. Denote $D_0 = D$.
3. For $k = 0, 1, \cdots, N(n) - 1$ where $N(n) = \frac{100l(n)}{\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2}$ is polynomial in n,
   (a) $\sigma_{D_k} \leftarrow \mathsf{Syn}^{|\mathsf{PSPACE}\rangle}(pk, D_k, s)$.
   (b) Run $\mathsf{Ver}^{\mathcal{O}}(pk, (s, \sigma_{D_k}))$.
   (c) Let D_{k+1} consist of all the query-answer pairs to $\mathcal{R}$ in item (b) and the pairs in D_k.
4. $j \xleftarrow{\$} \{0, 1, \ldots, N(n) - 1\}$.
5. Output $\phi = \phi_1 \otimes \phi_2$ where $\phi_i \leftarrow \mathsf{Syn}^{|\mathsf{PSPACE}\rangle}(pk, D_j, s)$ $(i = 1, 2)$.

Analysis of $\mathcal{A}^{\mathcal{O}}$ Now let's prove that $\mathcal{A}^{\mathcal{O}}$ outputs what we want. We will use the notations defined in the construction of $\mathcal{A}^{\mathcal{O}}$.

Theorem 6. *Given input $(pk, (s, \rho_s))$ generated by $\mathsf{KeyGen}^{\mathcal{O}}$ and $\mathsf{Mint}^{\mathcal{O}}$, $\mathcal{A}^{\mathcal{O}}$ outputs two alleged banknotes associated with the serial number s that will be accepted with high probability. Formally:*

$$\Pr\left[\mathsf{Ver}^{\mathcal{O}}(pk, (s, \phi_1)) \text{ accepts and } \mathsf{Ver}^{\mathcal{O}}(pk, (s, \phi_2)) \text{ accepts}\right] \geq 1.8\left(1 - \sqrt{1 - \delta_r + \epsilon}\right)^2 - 1,$$

where the probability is over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ (that is, the randomness of $\mathsf{KeyGen}^{\mathcal{O}}$ and $\mathsf{Mint}^{\mathcal{O}}$) and the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$.

Proof of Theorem 6. The proof will be divided into two parts. Informally speaking, in the first part, we will show that for every k, σ_{D_k} works well on the simulation, i.e. $\mathsf{V}^{\mathsf{PSPACE}}_{(pk,D_k,s)}(\sigma_{D_k})$ accepts with high probability; In the second part, we will show that for every k, if $\mathsf{Ver}^{\mathcal{O}}(pk, (s, \cdot))$ behaves far from $\mathsf{V}^{\mathsf{PSPACE}}_{(pk,D_k,s)}$ on input σ_{D_k}, then we make progress. Then we will combine the results to prove Theorem 6.

The first part The synthesizer Syn in Theorem 4 works well provided that good witness state for $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}$ exists. Our candidate for the good witness state is $\rho_s^{(t)}$ as it is accepted by $\mathsf{Ver}^{\mathcal{O}}$ with high probability by the definition of reusability. We begin by arguing that with high probability, our databases contain necessary information for running verification on $\rho_s^{(t)}$ and thus Ver can not distinguish whether it is interacting with random oracle $\mathcal{R}$ or the simulated one. Formally:

Claim 3. *Let $D_{\mathsf{KeyGen}}, D_{\mathsf{Mint}}$ be the query-answer pairs made during the generation of the input pk and (s, ρ_s) (that is, the execution of $\mathsf{KeyGen}^{\mathcal{O}}$ and $\mathsf{Mint}^{\mathcal{O}}$). Then*

$$\Pr\left[\mathsf{Ver}^{\mathcal{O}}(pk, (s, \rho_s^{(t)})) \text{ queries in } D_{\mathsf{Mint}} \cup D_{\mathsf{KeyGen}} - D\right] \leq \epsilon,$$

where the probability is over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ and the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$.

Proof. We only care about $l(n)$ query positions (those inside $D_{\mathsf{Mint}} \cup D_{\mathsf{KeyGen}}$) and we repeatedly sample $\lceil \frac{\ell}{\epsilon} \rceil$ times. Thus intuitively D should reveal all the positions we care about. Formally,

$$\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))\text{ queries in }D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D\right]$$

$$\leq \sum_{q\in D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}} \mathsf{Pr}\left[t=\min_j\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(j)}))\text{ queries }q\right]\right]$$
$$\leq \ell\cdot\frac{1}{\lceil\frac{\ell}{\epsilon}\rceil}$$
$$\leq \epsilon$$

where the probabilities are only over the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$ and we use the fact that t is picked uniformly random from $\{0,1,\ldots,\lceil\frac{\ell}{\epsilon}\rceil-1\}$, so it matches $\min_j\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(j)}))\text{ queries }q\right]$ (which may follow some distribution, but is independent of t anyway) with probability less or equal $\frac{1}{\lceil\frac{\ell}{\epsilon}\rceil}$.

After taking the randomness of $\mathcal{R}$ and the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ into account, we can get the claim.

□

The random oracle $\mathcal{R}$ can be implemented by on-the-fly simulation. Thus $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))$ can be implemented by simulating $\mathcal{R}$ with database $D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}\cup D$. If Ver doesn't make queries in $(D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}\cup D)\Delta D = D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D$, then it can not distinguish whether $\mathcal{R}$ is simulated with $D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}\cup D$ or D. That is, D is a good database to simulate the verification process on input $\rho_s^{(t)}$ if Ver doesn't make queries inside $D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D$. Thus the acceptance probability of $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D,s)}(\rho_s^{(t)})$ should be close to that of $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))$, which is high by the definition of reusability. On average, the performance of the simulation on input $\rho_s^{(t)}$ can only increase if we include more queries into the database. Thus for every k, $\rho_s^{(t)}$ should be a good witness state for $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}$. The intuition above is captured by the following claim.

Claim 4. *We use the same definition of D_{KeyGen} and D_{Mint} as in Claim 3. $\forall k\in[N(n)]$,*

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\rho_s^{(t)})\ \textit{accepts}\right]\geq\delta_r-\epsilon$$

where the probability is over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ and the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$.

Proof. This claim follows from Definition 4 and Claim 3. The following probabilities are over the

same randomness as the probability in the above claim.

$$\begin{aligned}
&\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right]\\
=&\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))\text{ accepts and queries in }D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D_k\right]\\
&+\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))\text{ accepts and never queries in }D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D_k\right]\\
\leq&\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))\text{ queries in }D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D\right]\\
&+\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\rho_s^{(t)})\text{ accepts and never queries in }D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}-D_k\right]\\
\leq&\epsilon+\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\rho_s^{(t)})\text{ accepts}\right]
\end{aligned}$$

where we use the fact that $D\subseteq D_k$ and the fact that we can use on-the-fly simulation to implement $\mathcal{R}$. As a result, $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))$ can also be seen as simulating $\mathcal{R}$ with $D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}}\cup D_k$, which is different from D_k only on $(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}}\cup D_k)\Delta D_k=D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}}-D_k$. Thus Ver can not distinguish whether $\mathcal{R}$ is simulated with D_k or $D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}\cup D_k$ if it never queries in $D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}}-D_k$. The above inequality can be rearranged as

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\rho_s^{(t)})\text{ accepts}\right]\geq\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right]-\epsilon\geq\delta_r-\epsilon.$$

□

Intuitively, from Claim 4, for a large fraction of $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}$, good witness state exists. Therefore, our synthesizer can find an "almost optimal" one. Formally:

Claim 5. *For every $k\in[N(n)]$,*

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k})\ \textit{accepts}\right]\geq 0.99\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2$$

where the probability is over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ and the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$.

Proof. The following probabilities are over the same randomness as the probability in the above claim unless otherwise stated.

Define $S:=\{(pk,D_k,s):\max_w\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(w)\text{ accepts}\right]\geq 1-\sqrt{1-\delta_r+\epsilon}\}$ where the probability is only over the randomness of V. Then by Claim 4 and averaging argument,

$$\mathsf{Pr}\left[(pk,D_k,s)\in S\right]\geq 1-\sqrt{1-\delta_r+\epsilon}$$

By Theorem 4, $\forall(pk,D_k,s)\in S$, $\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k})\text{ accepts}\right]\geq 0.99(1-\sqrt{1-\delta_r+\epsilon})$ where the probability is only over the randomness of V. Therefore,

$$\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k})\text{ accepts}\right]\geq 0.99\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2.$$

□

The second part We already know that σ_{D_k} is accepted by $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}$ with high probability. The next step is to associate the acceptance probability of $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}$ and that of $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\cdot))$ on σ_{D_k}. If the difference of these two terms is large, the simulation with D_k is not good enough. That is, $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k}))$ asks some important queries outside D_k. So in this case, D_{k+1} will contain more important queries and we make progress. Formally:

Claim 6. *We use the same notation as above. For every $k \in [N(n)]$,*

$$\begin{aligned}&\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \ \textit{accepts}\right] - \mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \ \textit{accepts}\right]\\ \leq&\mathbf{E}\left[|D_{k+1}\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right] - \mathbf{E}\left[|D_k\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right]\end{aligned}$$

where the probabilities and the expectations are over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ and the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$.

Proof. The following probabilities and expectations are over the same randomness of those in the above claim unless otherwise stated.

Similar as the arguments in Claim 4, $\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,s,D_k)}(\sigma_{D_k})$ and $\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k}))$ behave differently only when they make queries in $(D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}}\cup D_k)\Delta D_k = D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k$. Therefore,

$$\begin{aligned}&\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ accepts}\right]\\ =&\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ accepts and queries in } D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k\right]\\ &+\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ accepts and never queries in } D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k\right]\\ \leq&\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ queries in } D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k\right]\\ &+\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ accepts and never queries in } D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k\right]\\ =&\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ queries in } D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k\right]\\ &+\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ accepts and never queries in } D_{\mathsf{Mint}}\cup D_{\mathsf{KeyGen}} - D_k\right]\\ \leq&\mathsf{Pr}\left[(D_{k+1}-D_k)\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})\neq\emptyset\right] + \mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ accepts}\right]\\ \leq&\mathbf{E}\left[|(D_{k+1}-D_k)\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right] + \mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ accepts}\right]\end{aligned}$$

Note that $D_k \subseteq D_{k+1}$,

$$\mathbf{E}\left[|(D_{k+1}-D_k)\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right] = \mathbf{E}\left[|D_{k+1}\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right] - \mathbf{E}\left[|D_k\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right]$$

Therefore, the claim holds true. □

Now let's combine the above results to prove Theorem 6. The probabilities and expectations below are over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{O}}$ and the randomness of our adversary $\mathcal{A}^{\mathcal{O}}$ (thus over the randomness of t and j) unless otherwise stated.

By our construction and the union bound,

$$\begin{aligned}
&\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\phi_1)) \text{ accepts and } \mathsf{Ver}^{\mathcal{O}}(pk,(s,\phi_2)) \text{ accepts}\right]\\
\geq&2\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_j})) \text{ accepts}\right]-1\\
=&\frac{2}{N(n)}\sum_{k=0}^{N(n)-1}\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ accepts}\right]-1
\end{aligned}$$

From Claim 5 and Claim 6,

$$\begin{aligned}
&\frac{1}{N(n)}\sum_{k=0}^{N(n)-1}\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\sigma_{D_k})) \text{ accepts}\right]\\
\geq&\frac{1}{N(n)}\sum_{k=0}^{N(n)-1}\left(\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ accepts}\right]-\mathbf{E}\left[|D_{k+1}\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right]+\mathbf{E}\left[|D_k\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right]\right)\\
\geq&\frac{1}{N(n)}\sum_{k=0}^{N(n)-1}\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ accepts}\right]-\frac{1}{N(n)}\mathbf{E}\left[|D_{N(n)}\cap(D_{\mathsf{KeyGen}}\cup D_{\mathsf{Mint}})|\right]\\
\geq&\frac{1}{N(n)}\sum_{k=0}^{N(n)-1}\mathsf{Pr}\left[\mathsf{V}^{|\mathsf{PSPACE}\rangle}_{(pk,D_k,s)}(\sigma_{D_k}) \text{ accepts}\right]-\frac{l(n)}{N(n)}\\
\geq&0.99\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2-0.01\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2\\
\geq&0.9\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2
\end{aligned}$$

Therefore,

$$\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{O}}(pk,(s,\phi_1)) \text{ accepts and } \mathsf{Ver}^{\mathcal{O}}(pk,(s,\phi_2)) \text{ accepts}\right]\geq 1.8\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2-1,$$

which ends our proof of Theorem 6. □

Proof of Theorem 5. The proposed adversary $\mathcal{A}^{\mathcal{O}}$ is a valid attack because when $\epsilon=0.01,\delta_r=0.99$,

$$1.8\left(1-\sqrt{1-\delta_r+\epsilon}\right)^2-1\geq 1.8(1-0.2)^2-1\geq 0.1,$$

which is non-negligible. □

6 Extensions to Quantum Access

In this section, we will explore a special case where some algorithms can have quantum access to the random oracle. We consider reusable secure oracle-aided public key quantum money scheme $(\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle},\mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle},\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle})$. Formally:

Theorem 7. *Reusable and secure oracle-aided public key quantum money scheme* $(\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}, \mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}, \mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle})$ *does not exist where* $\mathcal{R}$ *is a random oracle.*

Without loss of generality, we can suppose the algorithms only make queries to the random oracle on input length $l(n)$ and they receive 1 bit output where l is a polynomial. (If they make queries to $\mathcal{R}$ on various input lengths, suppose the maximal input length is $l'(n)$, and then we can modify them so that their queries with input length $k(n)$ will be made on input length $l(n) = l'(n)+\log l'(n)$ where the first $k(n)$ bits stores the true query position, the middle $l'(n) - k(n)$ bits are 0, and the last $\log l'(n)$ bits indicates $k(n)$.) Ver makes $q(n)$ classical queries to $\mathcal{R}$. KeyGen and Mint make $q'(n)$ quantum queries to $\mathcal{R}$ in total. Denote the reusability and the security of the scheme as δ_r and δ_s respectively where $\delta_r = 1 - \mathsf{negl}(n), \delta_s = \mathsf{negl}(n)$. When it is clear from the context, we sometimes omit n.

It's worth noting that the attacker in Section 5 doesn't take advantage of the fact that KeyGen and Mint there can only make classical queries to $\mathcal{R}$. In fact, the same attacker works even when KeyGen and Mint can make quantum queries to $\mathcal{R}$ (with some modifications on the number of iterations). To be more specific, here is our construction of the attacker where $T(n)$, $N(n)$, the guarantee a and the threshold b of Syn will be determined later.

$\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ It takes as input a valid banknote (s, ρ_s) and public key pk, and behaves as follows.

1. **Test phase**: Let $t \xleftarrow{\$} \{0, \ldots, T(n) - 1\}$. Let $D = \emptyset$, $\rho_s^{(0)} = \rho_s$. Run the following t times. In i^{th} iteration,
   (a) $\rho_s^{(i)} \otimes |\mathbf{x}\rangle\langle\mathbf{x}| \leftarrow \mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk, s, \rho_s^{(i-1)})$.
   (b) Add query-answer pairs to $\mathcal{R}$ in item (a) into D.
2. **Update phase**: Let $j \xleftarrow{\$} \{0, 1, \ldots, N(n) - 1\}$. Let $D_0 = D$. Run the following j times. In k^{th} iteration,
   (a) $\sigma_{D_{k-1}} \leftarrow \mathsf{Syn}^{|\mathsf{PSPACE}\rangle}(pk, D_{k-1}, s)$.
   (b) Run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk, (s, \sigma_{D_{k-1}}))$.
   (c) Let D_k consist of all the query-answer pairs to $\mathcal{R}$ in item (b) and the pairs in D_{k-1}.
3. **Synthesize phase**: Output $\phi = \phi_1 \otimes \phi_2$ where $\phi_i \leftarrow \mathsf{Syn}^{|\mathsf{PSPACE}\rangle}(pk, D_j, s)$ $(i = 1, 2)$.

This description of $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ is actually equivalent to our adversary in Section 5. We move the line $j \xleftarrow{\$} \{0, 1, \ldots, N(n) - 1\}$ to the front because it will be easier to analyze.

What is left is to prove an analogue of Theorem 6. That is, the output states of $\mathcal{A}$ will be accepted with high probability.

Theorem 8. *Given input* $(pk, (s, \rho_s))$ *generated by* $\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}$ *and* $\mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}$, $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ *outputs two alleged banknotes associated with the serial number* s *that will be accepted with high probability. Formally:*

$$\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk, (s, \phi_1)) \text{ accepts and } \mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk, (s, \phi_2)) \text{ accepts}\right] \geq 1.8\left(1 - \sqrt{1 - \delta_r + \epsilon}\right)^2 - 1,$$

where the probability is over the randomness of $\mathcal{R}$, the randomness of the generation of the input for $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ (that is, the randomness of $\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}$ and $\mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}$) and the randomness of our adversary $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$.

Similar to Theorem 6, we will show that the verification on $\sigma_{D_j} \leftarrow \mathsf{Syn}^{|\mathsf{PSPACE}\rangle}(pk, D_j, s)$ accepts with high probability and then prove the theorem by union bound.

In Section 5, we crucially rely on the fact that whenever we make a mistake, we make progress in the sense that we recover a query inside $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$. However, now $\mathsf{KeyGen}, \mathsf{Mint}$ can make quantum queries. As a result, KeyGen and Mint could "touch" exponentially many positions. Fortunately, the compressed oralce technique introduced by Zhandry [Zha18] can be seen as a quantum analogue of recording queries into a database. Basically, if we run all the algorithms in the *purified view* and see the register containing the oracle in Fourier basis (labeled F), then all except polynomially many positions are $|\hat{0}\rangle$ after polynomially many quantum queries, and thus the register can be compressed using a unitary. In this work, in order to better mimic D_k and $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$ in Section 5, we take advantage of the fact that Ver only makes classical queries. To be more specific, we will maintain a register to store a database for all the classical queries and only record those non-$|\hat{0}\rangle$ positions outside the database into F. These two registers will be our analogue of D_k and $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$. We will elaborate this idea in Section 6.2.

6.1 A Purified View of the Algorithms

From Section 3.4, for any sequence of algorithms that only make queries to the random oracle on input length $l(n)$ and receive 1 bit output, we can analyze the output using a pure state that we obtain by running all the algorithms in the *purified view* instead. By *purified view*, we mean that we will purify the execution of the algorithms in the following way:

- We will introduce a register F that contains the truth table of the oracle. Before the execution of the first algorithm, it is initialized to a uniform superposition of all the possible truth table of the oracle, i.e. $|\hat{0}\rangle^{\otimes 2^l}$.
- Instead of quantum query to $\mathcal{R}$, we apply a unitary $U_Q : |x\rangle_{\mathsf{Q}} |y\rangle_{\mathsf{A}} |f\rangle_{\mathsf{F}} \to |x\rangle_{\mathsf{Q}} |y \oplus f(x)\rangle_{\mathsf{A}} |f\rangle_{\mathsf{F}}$ where Q stores the query position and A is for the answer bit. (The subscript Q in U_Q is for Quantum queries.)
- Instead of computational basis measurements, we apply $CNOT$ to copy it to a fresh ancilla.

Without loss of generality, we can suppose for any classical query to $\mathcal{R}$, the register for query answer is always set to $|0\rangle$ before the query. Notice that a classical query to $\mathcal{R}$ is equivalent to a computational basis measurement on the query position followed by a quantum query to $\mathcal{R}$. An extra computational basis measurement on the answer of the query won't change the view. So a classical query in the purified view can be treated as applying the unitary

$$U_C : |x\rangle_{\mathsf{Q}} |0\rangle_{\mathsf{A}} |f\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}} \to |x\rangle_{\mathsf{Q}} |f(x)\rangle_{\mathsf{A}} |f\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, f(x))\rangle_{\mathsf{D}_{\mathcal{R}}}$$

where $\mathsf{D}_{\mathcal{R}}$ is a register that we will use to purify the computational basis measurements (the last item in the above paragraph). By $|D_{\mathcal{R}}\rangle$, we mean $|(x_1, z_1), (x_2, z_2), \cdots (x_k, z_k)\rangle$ and $x_1, x_2, \cdots, x_k$ are not neccessary to be distinct but if $x_i = x_j$, then we have the guarantee that $z_i = z_j$. Here $\mathsf{D}_{\mathcal{R}}$ has enough space. That is, by $|(x_1, z_1), \cdots (x_k, z_k)\rangle$, we actually mean $|(x_1, z_1), \cdots (x_k, z_k), \perp, \cdots, \perp\rangle$

where $\perp$ is a special symbol that represents empty. All the database registers below also have enough space. (The subscript C in U_C is for Classical queries.)

Another convenient way to think of the purified view is to treat the execution of the algorithms as an interaction between two parties, the algorithm and the oracle. The oracle will maintain two private registers F and $\mathsf{D}_{\mathcal{R}}$ (and also some ancillas initialized to be $|0\rangle$). If the algorithm is allowed to make quantum queries to $\mathcal{R}$, the algorithm will submit QA to the oracle, and then the oracle will apply U_Q and send QA back to the algorithm. If the algorithm is only allowed to make classical queries, the algorithm will submit Q to the oracle, and then the oracle will put a fresh ancilla on A, apply U_C and send QA to the algorithm.

We will use $U_{\mathsf{KeyGen},n}, U_{\mathsf{Mint},n}, U_{\mathsf{Ver},n}$ and $U_{\mathsf{Syn},n}$ to denote the unitary corresponding to the purified version of KeyGen, Mint, Ver and Syn on security number n respectively. Then $U_{\mathsf{KeyGen},n}, U_{\mathsf{Mint},n}$ and $U_{\mathsf{Ver},n}$ are all in the form of preparing the first query and then repeatively answering the query by applying U_Q or U_C and preparing the next query (or the final output if there is no further query). In particular, we will write $U_{\mathsf{Ver},n}$ as $U_{q(n)}U_C U_{q(n)-1}\cdots U_C U_0$. We will omit the subscript n when it is clear from the context.

Let $U'_{\mathsf{Ver}} := U_q U_R U_{q-1}\cdots U_R U_0$ where U_R (the subscript R is for Recording) is a unitary that in addition to a classical query U_C, it records the query-answer pair into a database maintained by $\mathcal{A}$. That is,

$$U_R : |x\rangle_{\mathsf{Q}} |0\rangle_{\mathsf{A}} |D_{\mathcal{A}}\rangle_{\mathsf{D}_{\mathcal{A}}} |f\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}} \to |x\rangle_{\mathsf{Q}} |f(x)\rangle_{\mathsf{A}} |D_{\mathcal{A}}, (x, f(x))\rangle_{\mathsf{D}_{\mathcal{A}}} |f\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, f(x))\rangle_{\mathsf{D}_{\mathcal{R}}}$$

where $\mathsf{D}_{\mathcal{A}}$ is the register that stores the database maintained by $\mathcal{A}$. Again by $|D_{\mathcal{A}}\rangle$ and $|D_{\mathcal{R}}\rangle$, we mean a sequence of query-answer pairs where the query positions are not neccessary to be distinct but the pairs are consistent.

It's easy to see that U'_{Ver} corresponds to running $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ while the adversary records the query-answer pairs made by $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$.

Then in the purified view, $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ is the following (we will denote by $U_{\mathcal{A}}$):

1. Given input public key, serial number, the alleged banknote along with the register containing the truth table of the oracle, introduce a register T initialized to be $\frac{1}{\sqrt{T(n)}}\sum_{t=0}^{T(n)-1}|t\rangle_{\mathsf{T}}$ and introduce a register J initialized to be $\frac{1}{\sqrt{N(n)}}\sum_{j=0}^{N(n)-1}|j\rangle_{\mathsf{J}}$.

2. **Test phase**: Conditioned on the content in T is t, apply U'_{Ver} on the valid banknote (or the residual state of the valid banknote after several verifications) for t times in sequential. (Or equivalently apply unitary $U_{\mathsf{Test}} := \sum_{t=0}^{T(n)-1} U'^{t}_{\mathsf{Ver}} \otimes |t\rangle\langle t|_{\mathsf{T}}$ where U'^{t}_{Ver} means applying U'_{Ver} for t times.)

3. **Update phase**: Conditioned on the content in J is j, apply the following for j times:

   (a) Apply U_{Syn} on all the query-answer pairs we learn so far (i.e. the contents in $\mathsf{D}_{\mathcal{A}}$).

   (b) Apply U'_{Ver} on the state synthesized in item (a).

   (Or equivalently apply unitary $U_{\mathsf{Upd}} := \sum_{j=0}^{N(n)-1}(U'_{\mathsf{Ver}}U_{\mathsf{Syn}})^j \otimes |j\rangle\langle j|_{\mathsf{J}}$ where $(U'_{\mathsf{Ver}}U_{\mathsf{Syn}})^j$ means alternatively applying U_{Syn} and U'_{Ver} for j times.)

4. **Synthesize phase**: Apply U_{Syn} for two times on the query-answer pairs we learn.

Then the acceptance probability of the following algorithms on the corresponding states (taking the randomness of $\mathcal{R}$ into account) can be analyzed by running the following sequence of algorithms in the purified view.

$\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ The acceptance probability is the probability that

1. Run the algorithms $\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}$, $\mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}$ and $\mathcal{A}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ in a purified view sequentially where the input of KeyGen is the security parameter in unary notion, the input of Mint is the output register corresponding to secret key of KeyGen, and the input of $\mathcal{A}$ is the output register of Mint and the register for the public key.
2. Furthermore run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ in a purified view where the input of Ver is the register for the public key, the register for the serial number and the register for $\rho_s^{(t)}$ (It's inside working space register of $\mathcal{A}$).
3. Measure the register for outcome of the above $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ and obtain Accept.

$\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ Here $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ means running $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ where the queries to $\mathcal{R}$ is answered by the database D_j (all the query-answer pairs in $\mathsf{D}_{\mathcal{A}}$).

Define a unitary

$$U_D : |x\rangle_{\mathsf{Q}} |0\rangle_{\mathsf{A}} |D_j\rangle_{\mathsf{D}_{\mathcal{A}}} \to \begin{cases} |x\rangle_{\mathsf{Q}} |D_j(x)\rangle_{\mathsf{A}} |D_j,(x,D_j(x))\rangle_{\mathsf{D}_{\mathcal{A}}}, & x \in D_j \\ |x\rangle_{\mathsf{Q}} \sum_{z=0}^{1} \frac{1}{\sqrt{2}} |z\rangle_{\mathsf{A}} |D_j,(x,z)\rangle_{\mathsf{D}_{\mathcal{A}}}, & x \notin D_j \end{cases}$$

where by $|D_j\rangle$, we mean $|(x_1,z_1),(x_2,z_2),\cdots(x_k,z_k)\rangle$ and $x_1,x_2,\cdots,x_k$ are not neccessary to be distinct but if $x_i = x_j$, then we have the guarantee that $z_i = z_j$. By $x \in D_j$, we mean there exists z such that (x,z) is a pair in D_j and we will denote this z as $D_j(x)$. By $x \notin D_j$, we mean for all z, (x,z) is not a pair in D_j. (The subscript D in U_D is for simulating with Database.)

Then applying U_D is exactly answering the query x using D_j (If x is in the database, then answer the query using D_j; Otherwise, give an random answer while recording this query-answer pair into the database for later use). Thus the purified version of $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ is

$$U_{\mathsf{Sim}} := U_q U_D U_{q-1} \cdots U_D U_0.$$

So the acceptance probability of $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ is the probability that

1. Run the first step in the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$.
2. Furthermore run $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ in a purified view where the input of Ver is the register for the public key, the register for the serial number and the register for $\rho_s^{(t)}$ (The input is the same as the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$).
3. Measure the register for outcome of the above $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ and obtain Accept.

$\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ **on** σ_{D_j} The acceptance probability is the probability that

1. Run the first step in the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$.
2. Furthermore run $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ in a purified view where the input of Ver is the register for the public key, the register for the serial number and the register for σ_{D_j} (It's the first register of the output state of $\mathcal{A}$).
3. Measure the register for outcome of the above $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ and obtain Accept.

$\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ **on** σ_{D_j} The acceptance probability is the probability that

1. Run the first step in the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$.
2. Furthermore run $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ in a purified view where the input of Ver is the register for the public key, the register for the serial number and the register for σ_{D_j} (The input is the same as the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on σ_{D_j}).
3. Measure the register for outcome of the above $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ and obtain Accept.

6.2 Compress and Decompress

Intuitively, those $|\hat{0}\rangle$ positions in F is a uniform superposition of the range and it is unentangled with all other things, so it can be seen as independently choosing a value from the range uniformly at random, which is exactly what the simulation does. It is an analogue of those positions never asked during the sequence of algorithms in the purely classical query case.

In this subsection, we will show how to extract an analogue of D_k and $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$ from the pure state. Roughly speaking, the recorded classical queries is an analogue of D_k and we will compress the register F to extract those non-$|\hat{0}\rangle$ positions outside D_k to form our analogue of $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$.

As it's easier to write down and analyze the inverse operation of compress, we first give a formal description of decompress unitary Decomp. Recall that $\mathsf{D}_{\mathcal{R}}$ stores all the classical queries to $\mathcal{R}$. F is a register for the random function. Define

$$\mathsf{Decomp} : |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}} \to |f_0, f_1, \cdots, f_{2^l-1}\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}}$$

where $|D_F\rangle_{\mathsf{F}}$ can be written as a sequence of pairs $|(x_1,\widehat{y_1}),(x_2,\widehat{y_2}),\cdots,(x_{k'},\widehat{y_{k'}})\rangle_{\mathsf{F}}$, $|D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}}$ can be written as a sequence of pairs $|(x'_1,z_1),(x'_2,z_2),\cdots,(x'_k,z_k)\rangle$ and the input $|D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}}$ satisfies

- if $x'_i = x'_j$, then $z_i = z_j$;
- $x_1 < x_2 < \cdots < x_{k'}, \widehat{y_i} \neq \widehat{0}$;
- $\forall i,j, x_j \neq x'_i$;

and the output satisfies

- If $x'_j = i$, then $f_i = z_j$;
- If $x_j = i$, then $f_i = \widehat{y_j}$;

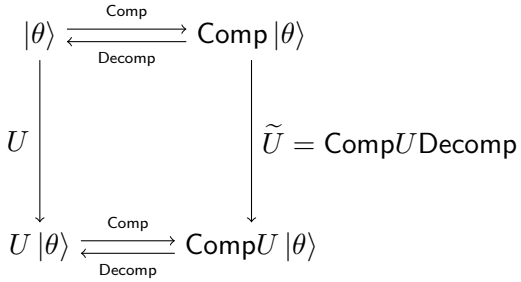
- If $\forall j, x_j \neq i, x'_j \neq i$, then $f_i = \widehat{0}$.

Roughly speaking, we fill $f_0, f_1, \cdots, f_{2^l-1}$ by looking at the pairs $(x'_1, z_1), (x'_2, z_2), \cdots, (x'_k, z_k)$ and $(x_1, \widehat{y_1}), (x_2, \widehat{y_2}), \cdots, (x_{k'}, \widehat{y_{k'}})$. And we fill all the remaining thing with $\hat{0}$.

Here our register F also have enough space. As our random function only has one-bit outputs, $z_1, z_2, \cdots, z_k, y_1, y_2, \cdots, y_{k'}$ are just 0 or 1. Recall that $|\hat{0}\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |\hat{1}\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. One can check that each two inputs in the above form are orthogonal and they are mapped to orthogonal outputs. So we can define the outputs of other inputs that are not in the above form so that Decomp is a unitary.

More Notations For simplicity, when we write $D_{\mathcal{R}}$, we mean $(x'_1, z_1), (x'_2, z_2), \cdots, (x'_k, z_k)$ that satisfies the first item of the input requirements above. By $x \in D_{\mathcal{R}}$, we mean $\exists i$, $x'_i = x$. By $D_{\mathcal{R}}(x)$ where $x \in D_{\mathcal{R}}$, we mean z_i where $x'_i = x$. When we write D_F, we mean a sequence $(x_1, \widehat{y_1}), (x_2, \widehat{y_2}), \cdots, (x_{k'}, \widehat{y_{k'}})$ that satisfies the second item of the input requirements above. By $x \in D_F$, we mean $\exists i$, $x_i = x$. By $D_F(x)$ where $x \in D_F$, we mean $\widehat{y_i}$ where $x_i = x$. By $\widehat{D_F(x)}$ where $x \in D_F$, we mean y_i where $x_i = x$. By $D_F - x$, we mean the sequence we obtain after deleting $x_i, \widehat{y_i}$ from the sequence D_F for every $x_i = x$. Define $D_{\mathcal{R}} \cap D_F = \{x : x \in D_{\mathcal{R}} \text{ and } x \in D_F\}$.

The inverse operation of the above unitary Decomp is compress, which can take our database $D_{\mathcal{R}}$ and the truth table in register F as inputs and compress them into two databases $D_{\mathcal{R}}$ and D_F. Define it as $\mathsf{Comp} := \mathsf{Decomp}^\dagger$. These two unitaries enable us to change our view between the decompressed one (a database for classical queries and a truth table) and the compressed one (a database for classical queries and another database). Here is a picture to illustrate this. $|\theta\rangle$ is an arbitrary state without compression. U is a unitary in the decompressed view (It takes a state without compression as input and outputs a state without compression). Then $\widetilde{U} := \mathsf{Comp}U\mathsf{Decomp}$ is a compressed view version of U (It takes a state after compression as input and outputs a state after compression). From now on, when we write unitary $\widetilde{\cdot}$, we mean it is in the compressed view.



Readers can treat $D_{\mathcal{R}}$ as an analogue of database D_k in Section 5 and treat D_F as an analogue of databases $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$ in Section 5. Roughly speaking, $D_{\mathcal{R}}$ stores our classical queries. The query positions in D_F are those asked by KeyGen and Mint, but not recorded in $D_{\mathcal{R}}$. We can understand this analogue better after taking a look at how the queries look like in the compressed view.

Recall the unitaries U_C, U_R and U_D from Section 6.1. They are for classical query, classical query while recording and simulated classical query respectively. Their compressed versions are the unitaries $\widetilde{U_C} := \mathsf{Comp}U_C\mathsf{Decomp}$, $\widetilde{U_R} := \mathsf{Comp}U_R\mathsf{Decomp}$ and $\widetilde{U_D} := \mathsf{Comp}U_D\mathsf{Decomp}$.

From the description of Decomp, we can get for $D_F \cap D_{\mathcal{R}} = \emptyset$,

$$\begin{aligned}
&\widetilde{U_C}(|x\rangle_{\mathsf{Q}} |0\rangle_{\mathsf{A}} |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}}) \\
&= \begin{cases} |x\rangle_{\mathsf{Q}} |D_{\mathcal{R}}(x)\rangle_{\mathsf{A}} |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, D_{\mathcal{R}}(x))\rangle_{\mathsf{D}_{\mathcal{R}}} & x \in D_{\mathcal{R}} \\ |x\rangle_{\mathsf{Q}} \frac{1}{\sqrt{2}} \sum_{z=0}^{1} |z\rangle_{\mathsf{A}} |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{R}}} & x \notin D_{\mathcal{R}}, x \notin D_F \\ |x\rangle_{\mathsf{Q}} \frac{1}{\sqrt{2}} \sum_{z=0}^{1} (-1)^{z\widehat{D_F(x)}} |z\rangle_{\mathsf{A}} |D_F - x\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{R}}} & x \notin D_{\mathcal{R}}, x \in D_F \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\widetilde{U_R}(|x\rangle_{\mathsf{Q}} |0\rangle_{\mathsf{A}} |D_{\mathcal{A}}\rangle_{\mathsf{D}_{\mathcal{A}}} |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}}) \\
&= \begin{cases} |x\rangle_{\mathsf{Q}} |D_{\mathcal{R}}(x)\rangle_{\mathsf{A}} |D_{\mathcal{A}}, (x, D_{\mathcal{R}}(x))\rangle_{\mathsf{D}_{\mathcal{A}}} |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, D_{\mathcal{R}}(x))\rangle_{\mathsf{D}_{\mathcal{R}}} & x \in D_{\mathcal{R}} \\ |x\rangle_{\mathsf{Q}} \frac{1}{\sqrt{2}} \sum_{z=0}^{1} |z\rangle_{\mathsf{A}} |D_{\mathcal{A}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{A}}} |D_F\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{R}}} & x \notin D_{\mathcal{R}}, x \notin D_F \\ |x\rangle_{\mathsf{Q}} \frac{1}{\sqrt{2}} \sum_{z=0}^{1} (-1)^{z\widehat{D_F(x)}} |z\rangle_{\mathsf{A}} |D_{\mathcal{A}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{A}}} |D_F - x\rangle_{\mathsf{F}} |D_{\mathcal{R}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{R}}} & x \notin D_{\mathcal{R}}, x \in D_F \end{cases}
\end{aligned}$$

Since U_D does not act on F and $\mathsf{D}_{\mathcal{R}}$, $\widetilde{U_D} = U_D$. Thus

$$\begin{aligned}
&\widetilde{U_D}(|x\rangle_{\mathsf{Q}} |0\rangle_{\mathsf{A}} |D_{\mathcal{A}}\rangle_{\mathsf{D}_{\mathcal{A}}}) \\
&= \begin{cases} |x\rangle_{\mathsf{Q}} |D_{\mathcal{A}}(x)\rangle_{\mathsf{A}} |D_{\mathcal{A}}, (x, D_{\mathcal{A}}(x))\rangle_{\mathsf{D}_{\mathcal{A}}}, & x \in D_{\mathcal{A}} \\ |x\rangle_{\mathsf{Q}} \sum_{z=0}^{1} \frac{1}{\sqrt{2}} |z\rangle_{\mathsf{A}} |D_{\mathcal{A}}, (x, z)\rangle_{\mathsf{D}_{\mathcal{A}}}, & x \notin D_{\mathcal{A}} \end{cases}
\end{aligned}$$

By the description of $\widetilde{U_C}$ and $\widetilde{U_R}$, whenever we ask a classical query to $x \in D_{\mathcal{R}}$, we answer it with our database $D_{\mathcal{R}}$ and record $(x, D_{\mathcal{R}}(x))$ for another time; whenever we ask a classical query to $x \notin D_{\mathcal{R}}$ and $x \notin D_F$, we answer it with a random z and record (x, z) in our database for later use; whenever we ask a classical query to $x \notin D_{\mathcal{R}}$ and $x \in D_F$, we actually copy the answer from the D_F, record it, and remove x from D_F. The above three cases is analogous to the classical on-the-fly simulation where the query to $\mathcal{R}$ inside D_k can be answered by D_k, the query to $\mathcal{R}$ inside $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$ should be answered by $D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k$ and we can give a random answer to the query to outside $(D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k) \cup D_k$. It is worth pointing out that $\widetilde{U_C}$ and $\widetilde{U_R}$ maintain the property that $D_{\mathcal{R}} \cap D_F$ is empty (analogous to $(D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k) \cap D_k = \emptyset$).

Furthermore, recall that $U_{\mathsf{Ver}} = U_q U_C U_{q-1} \cdots U_1 U_C U_0$ and U_i does not act on F or $\mathsf{D}_{\mathcal{R}}$. Thus $\widetilde{U_i} = U_i$, $\widetilde{U_{\mathsf{Ver}}} = \widetilde{U_q}\widetilde{U_C}\widetilde{U_{q-1}} \cdots \widetilde{U_1}\widetilde{U_C}\widetilde{U_0} = U_q \widetilde{U_C} U_{q-1} \cdots U_1 \widetilde{U_C} U_0$. Similarly, recall that the purified version of $\mathsf{Ver}^{D_j, |\mathsf{PSPACE}\rangle}$ is $U_{\mathsf{Sim}} = U_q U_D U_{q-1} \cdots U_D U_0$. Thus $\widetilde{U_{\mathsf{Sim}}} = U_q \widetilde{U_D} U_{q-1} \cdots \widetilde{U_D} U_0 = U_{\mathsf{Sim}}$ (Because $\widetilde{U_D} = U_D$).

6.3 Analysis of $\mathcal{A}^{\mathcal{R}, |\mathsf{PSPACE}\rangle}$

Here we analyze the acceptance probability of $\mathsf{Ver}^{\mathcal{R}, |\mathsf{PSPACE}\rangle}$ on the output of our $\mathcal{A}^{\mathcal{R}, |\mathsf{PSPACE}\rangle}$. We reuse our ideas in Section 5. Readers are encouraged to refer to our notation table in Appendix A when confused about the notations.

The following claim can be seen as an analogue of Claim 6 except that we work on a general state instead of solely analyzing σ_{D_k}. It basically says that when the behavior of $\mathsf{Ver}^{\mathcal{R}, |\mathsf{PSPACE}\rangle}$ (corresponding to $\widetilde{U_{\mathsf{Ver}}} = U_q \widetilde{U_C} \cdots \widetilde{U_C} U_0$) is far from a simulation $\mathsf{Ver}^{D, |\mathsf{PSPACE}\rangle}$ (corresponding to $\widetilde{U_{\mathsf{Sim}}} = U_q U_D \cdots U_D U_0$), then the number of pairs in F (analogous to $|D_{\mathsf{KeyGen}} \cup D_{\mathsf{Mint}} - D_k|$) will drop a lot after the verification. The intuition is that roughly speaking, $\widetilde{U_C}$ and U_D only behave

differently when given a query position $x \in D_F$, in which case x will be excluded from D_F after applying $\widetilde{U_C}$. So it results in a decrement of the number of pairs in F. Formally:

Claim 7. *Denote O to be the observable corresponding to the number of pairs in F (i.e. half of the nonempty length in F). To be more specific, $O = \sum_{D_F} |D_F| \, |D_F\rangle\langle D_F|_{\mathsf{F}}$ where $|D_F|$ is the number of pairs in D_F.*

For a state $|\phi\rangle$ in the following form (i.e. it's in the compressed view and the contents in $\mathsf{D}_{\mathcal{R}}$ and $\mathsf{D}_{\mathcal{A}}$ are the same),

$$|\phi\rangle = \sum_{\substack{pk,s,m,D,D_F,g \\ s.t.\ D\cap D_F=\emptyset}} \alpha_{pk,s,m,D,D_F,g} \, |pk\rangle_{\mathsf{Pk}} \, |s\rangle_{\mathsf{S}} \, |m\rangle_{\mathsf{M}} \, |D\rangle_{\mathsf{D}_{\mathcal{A}}} \, |D_F\rangle_{\mathsf{F}} \, |D\rangle_{\mathsf{D}_{\mathcal{R}}} \, |g\rangle_{\mathsf{G}} \, .$$

Let $\Pr\left[\widetilde{U_{\mathsf{Ver}}} \text{ accepts when running on } |\phi\rangle\right]$ be the acceptance probability of $\widetilde{U_{\mathsf{Ver}}}$ when the public key, the serial number and the alleged money state are in Pk, S and M respectively (It also equals to the acceptance probability of U_{Ver} on $\mathsf{Decomp}\,|\phi\rangle$ because $\widetilde{U_{\mathsf{Ver}}}\,|\phi\rangle = \mathsf{Comp}U_{\mathsf{Ver}}(\mathsf{Decomp}\,|\phi\rangle)$ and Comp does not act on the output bit).

Let $\Pr\left[\widetilde{U_{\mathsf{Sim}}} \text{ accepts when running on } |\phi\rangle\right]$ be the acceptance probability of $\widetilde{U_{\mathsf{Sim}}}$ when the public key, the serial number, the alleged money state and the database for simulating the random oracle are in $\mathsf{Pk}, \mathsf{S}, \mathsf{M}$ and $\mathsf{D}_{\mathcal{A}}$ respectively (It also equals to the acceptance probability of U_{Sim} on $\mathsf{Decomp}\,|\phi\rangle$).

$$\begin{aligned}
&\left|\Pr\left[\widetilde{U_{\mathsf{Ver}}} \text{ accepts when running on } |\phi\rangle\right] - \Pr\left[\widetilde{U_{\mathsf{Sim}}} \text{ accepts when running on } |\phi\rangle\right]\right| \\
\leq &\mathrm{TD}\left(\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Ver}}}\,|\phi\rangle\langle\phi|\,\widetilde{U_{\mathsf{Ver}}}^{\dagger}), \mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Sim}}}\,|\phi\rangle\langle\phi|\,\widetilde{U_{\mathsf{Sim}}}^{\dagger})\right) \\
\leq &6\sqrt{q\left(\mathrm{Tr}(O\,|\phi\rangle\langle\phi|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\,|\phi\rangle\langle\phi|\,\widetilde{U_{\mathsf{Ver}}}^{\dagger})\right)}
\end{aligned}$$

Proof. The first inequality follows immediately from the fact that we can measuring a qubit (not in $\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}$) of $\widetilde{U_{\mathsf{Ver}}}\,|\phi\rangle$ and $\widetilde{U_{\mathsf{Sim}}}\,|\phi\rangle$ to obtain whether they accept and the fact that we can not distinguish two states with probability greater than the their trace distance.

We will omit the registers when it's clear from the context. Recall that $\widetilde{U_{\mathsf{Ver}}} = U_q\widetilde{U_C}\cdots\widetilde{U_C}U_0$ and $\widetilde{U_{\mathsf{Sim}}} = U_qU_D\cdots U_DU_0$. Let U'_D be the same as U_D except that it uses the contents in $D_{\mathcal{R}}$ as database for simulation instead of the contents in $D_{\mathcal{A}}$. To be more specific,

$$U'_D(|x\rangle_{\mathsf{Q}}\,|0\rangle_{\mathsf{A}}\,|D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}}) = \begin{cases} |x\rangle_{\mathsf{Q}}\,|D_{\mathcal{R}}(x)\rangle_{\mathsf{A}}\,|D_{\mathcal{R}},(x,D_{\mathcal{R}}(x))\rangle_{\mathsf{D}_{\mathcal{R}}}\,, & x\in D_{\mathcal{R}} \\ |x\rangle_{\mathsf{Q}}\sum_{z=0}^{1}\frac{1}{\sqrt{2}}\,|z\rangle_{\mathsf{A}}\,|D_{\mathcal{R}},(x,z)\rangle_{\mathsf{D}_{\mathcal{R}}}\,, & x\notin D_{\mathcal{R}} \end{cases}$$

Define $|\phi_j\rangle = \widetilde{U_C}U_{j-1}\cdots\widetilde{U_C}U_0\,|\phi\rangle$ where $0\leq j\leq q$. In order to analyze the difference of $\widetilde{U_{\mathsf{Ver}}}$ and $\widetilde{U_{\mathsf{Sim}}}$ on $|\phi\rangle$ (that is, q true queries and q simulated queries), it's enough to analyze the difference

of one true query and one simulated query. Formally,

$$
\begin{aligned}
&\mathrm{TD}\left(\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Ver}}}\left|\phi\right\rangle\!\left\langle\phi\right|\widetilde{U_{\mathsf{Ver}}}^{\dagger}), \mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Sim}}}\left|\phi\right\rangle\!\left\langle\phi\right|\widetilde{U_{\mathsf{Sim}}}^{\dagger})\right)\\
=&\mathrm{TD}\left(\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(U_q\left|\phi_q\right\rangle\!\left\langle\phi_q\right|U_q^{\dagger}), \mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(U_qU'_D\cdots U'_DU_0\left|\phi_0\right\rangle\!\left\langle\phi_0\right|U_0^{\dagger}U_D'^{\dagger}\cdots U_D'^{\dagger}U_q^{\dagger})\right)\\
\leq&\sum_{j=0}^{q-1}\mathrm{TD}\left(\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(U_qU'_D\cdots U_{j+1}\left|\phi_{j+1}\right\rangle\!\left\langle\phi_{j+1}\right|U_{j+1}^{\dagger}U_D'^{\dagger}\cdots U_q^{\dagger}), \mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(U_qU'_D\cdots U_j\left|\phi_j\right\rangle\!\left\langle\phi_j\right|U_j^{\dagger}U_D'^{\dagger}\cdots U_q^{\dagger})\right)\\
\leq&\sum_{j=0}^{q-1}\mathrm{TD}\left(\left|\phi_{j+1}\right\rangle\!\left\langle\phi_{j+1}\right|, U'_DU_j\left|\phi_j\right\rangle\!\left\langle\phi_j\right|U_j^{\dagger}U_D'^{\dagger}\right)\\
=&\sum_{j=0}^{q-1}\left\|\widetilde{U_C}U_j\left|\phi_j\right\rangle\!\left\langle\phi_j\right|U_j^{\dagger}\widetilde{U_C}^{\dagger}-U'_DU_j\left|\phi_j\right\rangle\!\left\langle\phi_j\right|U_j^{\dagger}U_D'^{\dagger}\right\|_1
\end{aligned}
$$

where we use the fact that $|\phi\rangle$ have the same contents on $\mathsf{D}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{R}}$ and thus $\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Sim}}}\left|\phi\right\rangle\!\left\langle\phi\right|\widetilde{U_{\mathsf{Sim}}}^{\dagger})$ equals to $\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(U_qU'_D\cdots U'_DU_0\left|\phi_0\right\rangle\!\left\langle\phi_0\right|U_0^{\dagger}U_D'^{\dagger}\cdots U_D'^{\dagger}U_q^{\dagger})$.

For $0\leq j\leq q-1$, U_j prepares the $(j+1)^{th}$ query, thus we can write $U_j\left|\phi_j\right\rangle$ as follows,

$$
U_j\left|\phi_j\right\rangle=\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}}\alpha_{x,D_F,D_{\mathcal{R}},h}\left|x\right\rangle_{\mathsf{Q}}\left|0\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D}_{\mathcal{R}}}\left|h\right\rangle_{\mathsf{H}}
$$

where Q is the next query position, A is for the query answer and H contains all other registers including the public key, serial number, $\mathsf{D}_{\mathcal{A}}$, etc. We can divide these terms into two categories, one is $x\notin D_F$, and the other one is $x\in D_F$. That is, $U_j\left|\phi_j\right\rangle=\sqrt{1-\alpha}\left|u\right\rangle+\sqrt{\alpha}\left|v\right\rangle$ where $\alpha:=\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset,x\in D_F}}\left|\alpha_{x,D_F,D_{\mathcal{R}},h}\right|^2$,

$$
\sqrt{1-\alpha}\left|u\right\rangle=\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset,x\notin D_F}}\alpha_{x,D_F,D_{\mathcal{R}},h}\left|x\right\rangle_{\mathsf{Q}}\left|0\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D}_{\mathcal{R}}}\left|h\right\rangle_{\mathsf{H}}
$$

$$
\sqrt{\alpha}\left|v\right\rangle=\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset,x\in D_F}}\alpha_{x,D_F,D_{\mathcal{R}},h}\left|x\right\rangle_{\mathsf{Q}}\left|0\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D}_{\mathcal{R}}}\left|h\right\rangle_{\mathsf{H}}
$$

Recall that

$$
\begin{aligned}
&\widetilde{U_C}(\left|x\right\rangle_{\mathsf{Q}}\left|0\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D}_{\mathcal{R}}})\\
=&\begin{cases}
\left|x\right\rangle_{\mathsf{Q}}\left|D_{\mathcal{R}}(x)\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}},(x,D_{\mathcal{R}}(x))\right\rangle_{\mathsf{D}_{\mathcal{R}}} & x\in D_{\mathcal{R}}\\
\left|x\right\rangle_{\mathsf{Q}}\frac{1}{\sqrt{2}}\sum_{z=0}^{1}\left|z\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}},(x,z)\right\rangle_{\mathsf{D}_{\mathcal{R}}} & x\notin D_{\mathcal{R}},x\notin D_F\\
\left|x\right\rangle_{\mathsf{Q}}\frac{1}{\sqrt{2}}\sum_{z=0}^{1}(-1)^{z\widehat{D_F(x)}}\left|z\right\rangle_{\mathsf{A}}\left|D_F-x\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}},(x,z)\right\rangle_{\mathsf{D}_{\mathcal{R}}} & x\notin D_{\mathcal{R}},x\in D_F
\end{cases}
\end{aligned}
$$

So when $x \notin D_F$, $\widetilde{U_C}$ and U'_D act exactly the same. As a result, $\widetilde{U_C}\left|u\right\rangle = U'_D\left|u\right\rangle$. Therefore,

$$
\begin{aligned}
&\left\|\widetilde{U_C}U_j\left|\phi_j\right\rangle\left\langle\phi_j\right|U_j^\dagger\widetilde{U_C}^\dagger - U'_D U_j\left|\phi_j\right\rangle\left\langle\phi_j\right|U_j^\dagger U_D'^\dagger\right\|_1\\
=&\|\widetilde{U_C}\left((1-\alpha)\left|u\right\rangle\left\langle u\right| + \alpha\left|v\right\rangle\left\langle v\right| + \sqrt{\alpha(1-\alpha)}\left(\left|u\right\rangle\left\langle v\right| + \left|v\right\rangle\left\langle u\right|\right)\right)\widetilde{U_C}^\dagger\\
&- U'_D\left((1-\alpha)\left|u\right\rangle\left\langle u\right| + \alpha\left|v\right\rangle\left\langle v\right| + \sqrt{\alpha(1-\alpha)}\left(\left|u\right\rangle\left\langle v\right| + \left|v\right\rangle\left\langle u\right|\right)\right)U_D'^\dagger\|_1\\
=&\|\widetilde{U_C}\left(\alpha\left|v\right\rangle\left\langle v\right| + \sqrt{\alpha(1-\alpha)}\left(\left|u\right\rangle\left\langle v\right| + \left|v\right\rangle\left\langle u\right|\right)\right)\widetilde{U_C}^\dagger - U'_D\left(\alpha\left|v\right\rangle\left\langle v\right| + \sqrt{\alpha(1-\alpha)}\left(\left|u\right\rangle\left\langle v\right| + \left|v\right\rangle\left\langle u\right|\right)\right)U_D'^\dagger\|_1\\
\leq&2\left\|\alpha\left|v\right\rangle\left\langle v\right| + \sqrt{\alpha(1-\alpha)}\left(\left|u\right\rangle\left\langle v\right| + \left|v\right\rangle\left\langle u\right|\right)\right\|_1\\
\leq&2(\alpha + 2\sqrt{\alpha(1-\alpha)})\\
\leq&6\sqrt{\alpha}
\end{aligned}
$$

where we use properties of trace norm of matrices $\|A+B\|_1 \leq \|A\|_1 + \|B\|_1$, $\left\|UAU^\dagger\right\|_1 = \|A\|_1$ and $\|A\|_1 = \sum_i |\lambda_i|$ if $A = \sum_i \lambda_i \left|s_i\right\rangle\left\langle v_i\right|$ where $\left|s_i\right\rangle, \left|v_i\right\rangle$ are two bases.

Now let's associate α with other quantities. To be more precise, our goal is to bound α by the difference of the number of pairs in F on $\left|\phi_j\right\rangle$ and $\left|\phi_{j+1}\right\rangle = \widetilde{U_C}U_j\left|\phi_j\right\rangle$.

First, let's calculate the number of pairs in F on state $\left|\phi_j\right\rangle$.

Notice that U_j does not act on F, so it commutes with O. Then $\mathrm{Tr}(O\left|\phi_j\right\rangle\left\langle\phi_j\right|) = \mathrm{Tr}(OU_j\left|\phi_j\right\rangle\left\langle\phi_j\right|U_j^\dagger)$. Recall that

$$U_j\left|\phi_j\right\rangle = \sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} \alpha_{x,D_F,D_{\mathcal{R}},h}\left|x\right\rangle_{\mathsf{Q}}\left|0\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D_R}}\left|h\right\rangle_{\mathsf{H}}$$

In the summation, the terms are orthogonal to each other. Thus the expected number of pairs in F of $U_j\left|\phi_j\right\rangle$ is $\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} |D_F|\,|\alpha_{x,D_F,D_{\mathcal{R}},h}|^2$. So

$$\mathrm{Tr}(O\left|\phi_j\right\rangle\left\langle\phi_j\right|) = \sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} |D_F|\,|\alpha_{x,D_F,D_{\mathcal{R}},h}|^2$$

Next, let's calculate the number of pairs in F on state $\left|\phi_{j+1}\right\rangle$.

By definition, $\left|\phi_{j+1}\right\rangle = \widetilde{U_C}U_j\left|\phi_j\right\rangle$. Hence $\mathrm{Tr}(O\left|\phi_{j+1}\right\rangle\left\langle\phi_{j+1}\right|) = \mathrm{Tr}(O\widetilde{U_C}U_j\left|\phi_j\right\rangle\left\langle\phi_j\right|U_j^\dagger\widetilde{U_C}^\dagger)$. Notice that

$$\widetilde{U_C}U_j\left|\phi_j\right\rangle = \sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} \alpha_{x,D_F,D_{\mathcal{R}},h}\widetilde{U_C}(\left|x\right\rangle_{\mathsf{Q}}\left|0\right\rangle_{\mathsf{A}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D_R}})\left|h\right\rangle_{\mathsf{H}}$$

As $\widetilde{U_C}$ is a unitary, the terms in the above summation are orthogonal to each other.

By definition of $\widetilde{U_C}$, if we write $\widetilde{U_C}(|x\rangle_{\mathsf{Q}}|0\rangle_{\mathsf{A}}|D_F\rangle_{\mathsf{F}}|D_{\mathcal{R}}\rangle_{\mathsf{D}_{\mathcal{R}}})$ explicitly,

$$\begin{aligned}
&\widetilde{U_C}U_j|\phi_j\rangle\\
=&\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x\in D_{\mathcal{R}}}} \alpha_{x,D_F,D_{\mathcal{R}},h}|x\rangle_{\mathsf{Q}}|D_{\mathcal{R}}(x)\rangle_{\mathsf{A}}|D_F\rangle_{\mathsf{F}}|D_{\mathcal{R}},(x,D_{\mathcal{R}}(x))\rangle_{\mathsf{D}_{\mathcal{R}}}|h\rangle_{\mathsf{H}}\\
&+\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x\notin D_{\mathcal{R}}, x\notin D_F}} \alpha_{x,D_F,D_{\mathcal{R}},h}\sum_{z=0}^{1}\frac{1}{\sqrt{2}}|x\rangle_{\mathsf{Q}}|z\rangle_{\mathsf{A}}|D_F\rangle_{\mathsf{F}}|D_{\mathcal{R}},(x,z)\rangle_{\mathsf{D}_{\mathcal{R}}}|h\rangle_{\mathsf{H}}\\
&+\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x\in D_F}} \alpha_{x,D_F,D_{\mathcal{R}},h}\sum_{z=0}^{1}\frac{1}{\sqrt{2}}(-1)^{z\widehat{D_F(x)}}|x\rangle_{\mathsf{Q}}|z\rangle_{\mathsf{A}}|D_F-x\rangle_{\mathsf{F}}|D_{\mathcal{R}},(x,z)\rangle_{\mathsf{D}_{\mathcal{R}}}|h\rangle_{\mathsf{H}}
\end{aligned}$$

Thus the expected number of pairs in F of $\widetilde{U_C}U_j|\phi_j\rangle$ (i.e. $\mathrm{Tr}(O\widetilde{U_C}U_j|\phi_j\rangle\langle\phi_j|U_j^\dagger\widetilde{U_C}^\dagger)$) is

$$\begin{aligned}
&\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x\notin D_F}} |D_F|\,|\alpha_{x,D_F,D_{\mathcal{R}},h}|^2+\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x\in D_F}} |D_F-x|\,|\alpha_{x,D_F,D_{\mathcal{R}},h}|^2\\
=&\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} |D_F|\,|\alpha_{x,D_F,D_{\mathcal{R}},h}|^2-\sum_{\substack{x,D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x\in D_F}} |\alpha_{x,D_F,D_{\mathcal{R}},h}|^2\\
=&\mathrm{Tr}(O|\phi_j\rangle\langle\phi_j|)-\alpha
\end{aligned}$$

That is, $\alpha=\mathrm{Tr}(O|\phi_j\rangle\langle\phi_j|)-\mathrm{Tr}(O|\phi_{j+1}\rangle\langle\phi_{j+1}|)$. As a result, roughly speaking, the difference of one true query and one simulated query is bounded by the decrement of number of pairs in F after one query. That is,

$$\left\|\widetilde{U_C}U_j|\phi_j\rangle\langle\phi_j|U_j^\dagger\widetilde{U_C}^\dagger-U_D'U_j|\phi_j\rangle\langle\phi_j|U_j^\dagger U_D'^\dagger\right\|_1\le 6\sqrt{\mathrm{Tr}(O|\phi_j\rangle\langle\phi_j|)-\mathrm{Tr}(O|\phi_{j+1}\rangle\langle\phi_{j+1}|)}$$

Combining the above equations, we can obtain for $\widetilde{U_{\mathsf{Ver}}}$ and $\widetilde{U_{\mathsf{Sim}}}$ which make q queries,

$$\begin{aligned}
&\mathrm{TD}\left(\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Ver}}}|\phi\rangle\langle\phi|\widetilde{U_{\mathsf{Ver}}}^\dagger),\mathrm{Tr}_{\mathsf{FD}_{\mathcal{A}}\mathsf{D}_{\mathcal{R}}}(\widetilde{U_{\mathsf{Sim}}}|\phi\rangle\langle\phi|\widetilde{U_{\mathsf{Sim}}}^\dagger)\right)\\
\le&\sum_{j=0}^{q-1}6\sqrt{\mathrm{Tr}(O|\phi_j\rangle\langle\phi_j|)-\mathrm{Tr}(O|\phi_{j+1}\rangle\langle\phi_{j+1}|)}\\
\le&6\sqrt{q\sum_{j=0}^{q-1}(\mathrm{Tr}(O|\phi_j\rangle\langle\phi_j|)-\mathrm{Tr}(O|\phi_{j+1}\rangle\langle\phi_{j+1}|))}\ \text{(Cauchy–Schwarz inequality)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O|\phi\rangle\langle\phi|)-\mathrm{Tr}(O\widetilde{U_C}U_{q-1}\cdots\widetilde{U_C}U_0|\phi\rangle\langle\phi|U_0^\dagger\widetilde{U_C}^\dagger\cdots U_{q-1}^\dagger\widetilde{U_C}^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O|\phi\rangle\langle\phi|)-\mathrm{Tr}(OU_q\widetilde{U_C}\cdots\widetilde{U_C}U_0|\phi\rangle\langle\phi|U_0^\dagger\widetilde{U_C}^\dagger\cdots\widetilde{U_C}^\dagger U_q^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O|\phi\rangle\langle\phi|)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}|\phi\rangle\langle\phi|\widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}
\end{aligned}$$

where we use again the fact that U_q does not act on F and thus O commutes with U_q.

□

The next claim is an analogue of Claim 4. Basically, it argues that on average, $\rho_s^{(t)}$ is a good witness state for simulation with database D_j even when KeyGen and Mint can make quantum queries to $\mathcal{R}$.

The intuition of the proof is the following: from Section 6.1, the difference between the acceptance probabilities of $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ and $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ is the difference of running $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}$ and $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ in the purified view on the same state (i.e. the difference of applying U_{Ver} and U_{Sim} on the same state), which can be transformed to the compressed view and by Claim 7 can be bounded by the difference of the number of pairs in F before and after the real verification. Roughly speaking, the decrement of the pairs is the number of pairs in D_F asked during the verification. But we randomize t, so running another verification on $\rho_s^{(t)}$ should not decrease the number of pairs in F too much on average. Formally:

Claim 8.

$$\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\ \textit{accepts}\right]-\Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\ \textit{accepts}\right]\right|\le 6\sqrt{\frac{q(n)q'(n)}{T(n)}}$$

where the probabilities are taken over the randomness of $\mathcal{R}$, the randomness of the inputs to the adversary and the randomness of the adversary (so t and j are both randomized).

Proof. From Section 6.1, these two probabilities can be analyzed by running a sequence of algorithms in the purified view. As $\mathsf{CompDecomp}=I$, we can insert Comp and Decomp between the algorithms. So these two probabilities can also be analyzed by running the sequence of algorithms in the compressed view.

Let $|\phi\rangle$ be the whole pure state we obtain by running the first step in the compressed view in the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ in Section 6.1 (i.e. applying the unitaries $\widetilde{U_{\mathsf{KeyGen}}}=\mathsf{Comp}U_{\mathsf{KeyGen}}\mathsf{Decomp}$, $\widetilde{U_{\mathsf{Mint}}}=\mathsf{Comp}U_{\mathsf{Mint}}\mathsf{Decomp}$ and $\widetilde{U_{\mathcal{A}}}=\mathsf{Comp}U_{\mathcal{A}}\mathsf{Decomp}$ to the state $|1^n\rangle|\emptyset\rangle_{\mathsf{D}_{\mathcal{A}}}|\emptyset\rangle_{\mathsf{F}}|\emptyset\rangle_{\mathsf{D}_{\mathcal{R}}}$ along with enough ancillas). Let M be the register corresponding to $\rho_s^{(t)}$.

It's easy to see that every classical query is recorded by the adversary. That is, $|\phi\rangle$ has the same contents in $\mathsf{D}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{R}}$. From Claim 7,

$$\begin{aligned}&\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\ \text{accepts}\right]-\Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\ \text{accepts}\right]\right|\\ \le&6\sqrt{q\left(\mathrm{Tr}(O|\phi\rangle\langle\phi|)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}|\phi\rangle\langle\phi|\widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}\end{aligned}$$

Denote $\widetilde{U_{\mathsf{Upd}}}$ to be unitary that describes our update phase in the compressed view. Formally, $\widetilde{U_{\mathsf{Upd}}}=\mathsf{Comp}U_{\mathsf{Upd}}\mathsf{Decomp}$ where U_{Upd} is the unitary that describes the update phase of $\mathcal{A}$ in Section 6.1. (That is, $U_{\mathsf{Upd}}=\sum_{j=0}^{N(n)-1}(U'_{\mathsf{Ver}}U_{\mathsf{Syn}})^j\otimes|j\rangle\langle j|_{\mathsf{J}}$ where U'_{Ver} is running on the state synthesized by U_{Syn}.) Then $\widetilde{U_{\mathsf{Upd}}}=\sum_{j=0}^{N(n)-1}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j\otimes|j\rangle\langle j|_{\mathsf{J}}$.

Let $|\psi\rangle$ be the whole pure state we obtain by running the first step in the compressed view in the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ until the end of the **test** phase of $\mathcal{A}$. That is to say, $|\phi\rangle=\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Upd}}}|\psi\rangle$. Notice that $\widetilde{U_{\mathsf{Syn}}}=U_{\mathsf{Syn}}$ only reads the public key, the serial number and

acts on $\mathsf{D}_{\mathcal{A}}$ and some fresh ancillas. So $\widetilde{U_{\mathsf{Syn}}}$ commutes with O and $\widetilde{U_{\mathsf{Ver}}}$. (Recall that $\widetilde{U_{\mathsf{Ver}}}$ is the one that does not records queries for $\mathcal{A}$. That is, it does not acts on $\mathsf{D}_{\mathcal{A}}$.) Thus

$$\begin{aligned}
&\mathrm{Tr}(O\left|\phi\right\rangle\!\left\langle\phi\right|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\left|\phi\right\rangle\!\left\langle\phi\right|\widetilde{U_{\mathsf{Ver}}}^{\dagger})\\
=&\mathrm{Tr}(O\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Upd}}}^{\dagger}\widetilde{U_{\mathsf{Syn}}}^{\dagger}\widetilde{U_{\mathsf{Syn}}}^{\dagger}) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Upd}}}^{\dagger}\widetilde{U_{\mathsf{Syn}}}^{\dagger}\widetilde{U_{\mathsf{Syn}}}^{\dagger}\widetilde{U_{\mathsf{Ver}}}^{\dagger})\\
=&\mathrm{Tr}(O\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Upd}}}^{\dagger}) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Upd}}}^{\dagger}\widetilde{U_{\mathsf{Ver}}}^{\dagger})
\end{aligned}$$

The next step is to bound the decrement of the pairs in F during the verification on $\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle$. However, the update phase between $\widetilde{U_{\mathsf{Ver}}}$ and the randomized number of verifications on ρ_s in the test phase may bring some trouble. So we give the following lemma, which is the counterpart of the fact that in the classical case, the number of queries in $D_{\mathsf{Mint}} \cup D_{\mathsf{KeyGen}} - D_k$ asked during a verification is no more than the number of queries in $D_{\mathsf{Mint}} \cup D_{\mathsf{KeyGen}} - D$ asked during a verification.

Lemma 6. *We use the same notation as above. Then*

$$\begin{aligned}
&\mathrm{Tr}(O\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Upd}}}^{\dagger}) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Upd}}}^{\dagger}\widetilde{U_{\mathsf{Ver}}}^{\dagger})\\
\leq&\mathrm{Tr}(O\left|\psi\right\rangle\!\left\langle\psi\right|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\left|\psi\right\rangle\!\left\langle\psi\right|\widetilde{U_{\mathsf{Ver}}}^{\dagger})
\end{aligned}$$

Proof. We first show that for any state

$$\left|\varphi\right\rangle = \sum_{\substack{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} \alpha_{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h}\left|x_1\right\rangle_{\mathsf{Q}_1}\left|0\right\rangle_{\mathsf{A}_1}\left|x_2\right\rangle_{\mathsf{Q}_2}\left|0\right\rangle_{\mathsf{A}_2}\left|D_{\mathcal{A}}\right\rangle_{\mathsf{D}_{\mathcal{A}}}\left|D_F\right\rangle_{\mathsf{F}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D}_{\mathcal{R}}}\left|h\right\rangle_{\mathsf{H}},$$

we have the inequality

$$\mathrm{Tr}(O\widetilde{U_R}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_R}^{\dagger}) - \mathrm{Tr}(O\widetilde{U_C}\widetilde{U_R}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_R}^{\dagger}\widetilde{U_C}^{\dagger}) \leq \mathrm{Tr}(O\left|\varphi\right\rangle\!\left\langle\varphi\right|) - \mathrm{Tr}(O\widetilde{U_C}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_C}^{\dagger})$$

where $\widetilde{U_R}$ acts on $\mathsf{Q}_1, \mathsf{A}_1, \mathsf{D}_{\mathcal{A}}, \mathsf{F}$ and $\mathsf{D}_{\mathcal{R}}$ while $\widetilde{U_C}$ acts on $\mathsf{Q}_2, \mathsf{A}_2, \mathsf{F}$ and $\mathsf{D}_{\mathcal{R}}$.

In fact, from the same argument in Claim 7, $\mathrm{Tr}(O\left|\varphi\right\rangle\!\left\langle\varphi\right|) - \mathrm{Tr}(O\widetilde{U_C}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_C}^{\dagger})$ is exactly the probability that we get outcome (x_2, D_F) such that $x_2 \in D_F$ when we measure the registers Q_2 and D_F on state $\left|\varphi\right\rangle$. Thus

$$\mathrm{Tr}(O\left|\varphi\right\rangle\!\left\langle\varphi\right|) - \mathrm{Tr}(O\widetilde{U_C}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_C}^{\dagger}) = \sum_{\substack{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x_2\in D_F}} \left|\alpha_{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h}\right|^2.$$

Similarly, $\mathrm{Tr}(O\widetilde{U_R}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_R}^{\dagger}) - \mathrm{Tr}(O\widetilde{U_C}\widetilde{U_R}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_R}^{\dagger}\widetilde{U_C}^{\dagger})$ is exactly the probability that we get outcome (x_1, D_F) such that $x_1 \in D_F$ when we measure the registers Q_1 and D_F on state $\widetilde{U_R}\left|\varphi\right\rangle$. Notice that we can write $\widetilde{U_R}\left|\varphi\right\rangle$ as

$$\sum_{\substack{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset}} \alpha_{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h}\left|x_2\right\rangle_{\mathsf{Q}_2}\left|0\right\rangle_{\mathsf{A}_2}\left|x_1\right\rangle_{\mathsf{Q}_1} U_{x_1,D_F}(\left|0\right\rangle_{\mathsf{A}_1}\left|D_{\mathcal{A}}\right\rangle_{\mathsf{D}_{\mathcal{A}}}\left|D_{\mathcal{R}}\right\rangle_{\mathsf{D}_{\mathcal{R}}})\left|D_F - x_1\right\rangle_{\mathsf{F}}\left|h\right\rangle_{\mathsf{H}}$$

where U_{x_1,D_F} is a unitary that depends on x_1, D_F. Thus

$$\mathrm{Tr}(O\widetilde{U_R}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_R}^{\dagger}) - \mathrm{Tr}(O\widetilde{U_C}\widetilde{U_R}\left|\varphi\right\rangle\!\left\langle\varphi\right|\widetilde{U_R}^{\dagger}\widetilde{U_C}^{\dagger}) = \sum_{\substack{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h\\ \text{s.t. } D_F\cap D_{\mathcal{R}}=\emptyset, x_2\in D_F - x_1}} \left|\alpha_{x_1,x_2,D_{\mathcal{A}},D_F,D_{\mathcal{R}},h}\right|^2.$$

As a result, $\mathrm{Tr}(O\widetilde{U_R}\left|\varphi\right\rangle\left\langle\varphi\right|\widetilde{U_R}^\dagger)-\mathrm{Tr}(O\widetilde{U_C}\widetilde{U_R}\left|\varphi\right\rangle\left\langle\varphi\right|\widetilde{U_R}^\dagger\widetilde{U_C}^\dagger)\leq\mathrm{Tr}(O\left|\varphi\right\rangle\left\langle\varphi\right|)-\mathrm{Tr}(O\widetilde{U_C}\left|\varphi\right\rangle\left\langle\varphi\right|\widetilde{U_C}^\dagger)$. That is to say, an extra query $\widetilde{U_R}$ on other part of the state can only decrease the chance of making a bad query in $\widetilde{U_C}$ because that extra query can only make the set of bad queries smaller.

$\widetilde{U_{\mathsf{Ver}}}$ and $\widetilde{U_{\mathsf{Upd}}}$ are composed of $\widetilde{U_C}$ and $\widetilde{U_R}$. In fact, the above argument can also be extended to $\widetilde{U_{\mathsf{Ver}}}$ and $\widetilde{U_{\mathsf{Upd}}}$ to capture our intuition that $\widetilde{U_{\mathsf{Upd}}}$ can only decrease the number of bad queries made during $\widetilde{U_{\mathsf{Ver}}}$ because $\widetilde{U_{\mathsf{Upd}}}$ can only make the set of bad queries smaller. We will first show a fixed number of iterations of update phase can only decrease the number of bad queries made during $\widetilde{U_{\mathsf{Ver}}}$ and then show it holds for $\widetilde{U_{\mathsf{Upd}}}$. Formally,

Note that we can write $\left|\psi\right\rangle$ as $\left|\psi_0\right\rangle\otimes\frac{1}{\sqrt{N(n)}}\sum_{j=0}^{N(n)-1}\left|j\right\rangle_{\mathsf{J}}$. Then

$$\widetilde{U_{\mathsf{Upd}}}\left|\psi\right\rangle=\frac{1}{\sqrt{N(n)}}\sum_{j=0}^{N(n)-1}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j(\left|\psi_0\right\rangle)\left|j\right\rangle_{\mathsf{J}}$$

For any $j\geq 1$, for unitary $\widetilde{U'_{\mathsf{Ver}}}$ that acts on the synthesized state and records the query for $\mathcal{A}$, and unitary $\widetilde{U_{\mathsf{Ver}}}$ that acts on $\rho_s^{(t)}$, denote $\left|w\right\rangle=\widetilde{U_{\mathsf{Syn}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^{j-1}\left|\psi_0\right\rangle$. Let $U_i^{(1)}$ prepare the next query for $\rho_s^{(t)}$ and $U_i^{(2)}$ prepare the next query for the synthesized state. For every i, denote $U_{i,C}^{(1)}=U_i^{(1)}\widetilde{U_C}\cdots U_0^{(1)}$ and $U_{i,R}^{(2)}=U_i^{(2)}\widetilde{U_R}\cdots U_0^{(2)}$. Repetitively applying the above inequality, we can get that

$$\begin{aligned}
&\mathrm{Tr}(O(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j\left|\psi_0\right\rangle\left\langle\psi_0\right|(\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)^j)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j\left|\psi_0\right\rangle\left\langle\psi_0\right|(\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)^j\widetilde{U_{\mathsf{Ver}}}^\dagger)\\
=&\mathrm{Tr}(OU_{q,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q,R}^{(2)}}^\dagger)-\mathrm{Tr}(OU_{q,C}^{(1)}U_{q,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q,R}^{(2)}}^\dagger{U_{q,C}^{(1)}}^\dagger)\\
=&\mathrm{Tr}(OU_q^{(2)}\widetilde{U_R}U_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger\widetilde{U_R}^\dagger{U_q^{(2)}}^\dagger)\\
&-\mathrm{Tr}(OU_q^{(1)}U_q^{(2)}\widetilde{U_C}\widetilde{U_R}U_{q-1,C}^{(1)}U_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger{U_{q-1,C}^{(1)}}^\dagger\widetilde{U_R}^\dagger\widetilde{U_C}^\dagger{U_q^{(2)}}^\dagger{U_q^{(1)}}^\dagger)\\
=&\mathrm{Tr}(O\widetilde{U_R}U_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger\widetilde{U_R}^\dagger)-\mathrm{Tr}(O\widetilde{U_C}\widetilde{U_R}U_{q-1,C}^{(1)}U_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger{U_{q-1,C}^{(1)}}^\dagger\widetilde{U_R}^\dagger\widetilde{U_C}^\dagger)\\
\leq&\mathrm{Tr}(OU_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger)-\mathrm{Tr}(O\widetilde{U_C}U_{q-1,C}^{(1)}U_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger{U_{q-1,C}^{(1)}}^\dagger\widetilde{U_C}^\dagger)\\
=&\mathrm{Tr}(OU_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger)-\mathrm{Tr}(OU_{q,C}^{(1)}U_{q-1,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{q-1,R}^{(2)}}^\dagger{U_{q,C}^{(1)}}^\dagger)
\end{aligned}$$

That is, we can delete a query for the synthesized state without decreasing the number of bad queries made during $\widetilde{U_{\mathsf{Ver}}}$ on $\rho_s^{(t)}$. Repetitively moving the queries for the synthesized state, we can get that

$$\begin{aligned}
&\mathrm{Tr}(O(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j\left|\psi_0\right\rangle\left\langle\psi_0\right|(\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)^j)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j\left|\psi_0\right\rangle\left\langle\psi_0\right|(\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)^j\widetilde{U_{\mathsf{Ver}}}^\dagger)\\
\leq&\mathrm{Tr}(OU_{0,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{0,R}^{(2)}}^\dagger)-\mathrm{Tr}(OU_{q,C}^{(1)}U_{0,R}^{(2)}\left|w\right\rangle\left\langle w\right|{U_{0,R}^{(2)}}^\dagger{U_{q,C}^{(1)}}^\dagger)\\
=&\mathrm{Tr}(O\left|w\right\rangle\left\langle w\right|)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\left|w\right\rangle\left\langle w\right|\widetilde{U_{\mathsf{Ver}}}^\dagger)
\end{aligned}$$

where we use the fact that $U_i^{(1)}$ commutes with $U_j^{(2)}$ and $\widetilde{U_C}$, $U_i^{(2)}$ commutes with $\widetilde{U_R}$, and both $U_i^{(1)}$ and $U_j^{(2)}$ commutes with O.

We successfully removed one $\widetilde{U'_{\mathsf{Ver}}}$ on the synthesized state. We can do it until we remove all of $\widetilde{U'_{\mathsf{Ver}}}$ on the synthesized state. That is,

$$
\begin{aligned}
&\mathrm{Tr}(O(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^j) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^j \widetilde{U_{\mathsf{Ver}}}^\dagger)\\
\leq&\mathrm{Tr}(O\widetilde{U_{\mathsf{Syn}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^{j-1} |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^{j-1} \widetilde{U_{\mathsf{Syn}}}^\dagger)\\
&- \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^{j-1} |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^{j-1} \widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U_{\mathsf{Ver}}}^\dagger)\\
=&\mathrm{Tr}(O(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^{j-1} |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^{j-1}) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^{j-1} |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^{j-1} \widetilde{U_{\mathsf{Ver}}}^\dagger)\\
\leq&\cdots\\
\leq&\mathrm{Tr}(O |\psi_0\rangle\langle\psi_0|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}} |\psi_0\rangle\langle\psi_0| \widetilde{U_{\mathsf{Ver}}}^\dagger)
\end{aligned}
$$

where we use the fact that $\widetilde{U_{\mathsf{Syn}}}$ uses the database in $D_{\mathcal{A}}$ and $\widetilde{U_{\mathsf{Ver}}}$ does not record query for $\mathcal{A}$. Thus $\widetilde{U_{\mathsf{Syn}}}$ and $\widetilde{U_{\mathsf{Ver}}}$ commute. $\widetilde{U_{\mathsf{Syn}}}$ does not act on F and $\mathsf{D}_{\mathcal{R}}$, so it commutes with O.

Finally, a randomized number of iterations of update phase can not increase the number of bad queries made during $\widetilde{U_{\mathsf{Ver}}}$. As both $\widetilde{U_{\mathsf{Ver}}}$ and O do not act on register J, tracing out J after $\widetilde{U_{\mathsf{Upd}}}$ does not change the quantity. Thus

$$
\begin{aligned}
&\mathrm{Tr}(O\widetilde{U_{\mathsf{Upd}}} |\psi\rangle\langle\psi| \widetilde{U_{\mathsf{Upd}}}^\dagger) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Upd}}} |\psi\rangle\langle\psi| \widetilde{U_{\mathsf{Upd}}}^\dagger \widetilde{U_{\mathsf{Ver}}}^\dagger)\\
=&\frac{1}{N(n)}\sum_{j=0}^{N(n)-1}\left(\mathrm{Tr}(O(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^j) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}(\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}})^j |\psi_0\rangle\langle\psi_0| (\widetilde{U_{\mathsf{Syn}}}^\dagger \widetilde{U'_{\mathsf{Ver}}}^\dagger)^j \widetilde{U_{\mathsf{Ver}}}^\dagger)\right)\\
\leq&\mathrm{Tr}(O |\psi_0\rangle\langle\psi_0|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}} |\psi_0\rangle\langle\psi_0| \widetilde{U_{\mathsf{Ver}}}^\dagger)\\
=&\mathrm{Tr}(O |\psi\rangle\langle\psi|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}} |\psi\rangle\langle\psi| \widetilde{U_{\mathsf{Ver}}}^\dagger)
\end{aligned}
$$

□

Now let's move on to bound the decrement of the pairs in F during the verification. From Lemma 6,

$$
\begin{aligned}
&\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right] - \Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right]\right|\\
\leq&6\sqrt{q\left(\mathrm{Tr}(O\widetilde{U_{\mathsf{Upd}}} |\psi\rangle\langle\psi| \widetilde{U_{\mathsf{Upd}}}^\dagger) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Upd}}} |\psi\rangle\langle\psi| \widetilde{U_{\mathsf{Upd}}}^\dagger \widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}\\
\leq&6\sqrt{q\left(\mathrm{Tr}(O |\psi\rangle\langle\psi|) - \mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}} |\psi\rangle\langle\psi| \widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O |\psi\rangle\langle\psi|) - \mathrm{Tr}(O\widetilde{U'_{\mathsf{Ver}}} |\psi\rangle\langle\psi| \widetilde{U'_{\mathsf{Ver}}}^\dagger)\right)}
\end{aligned}
$$

where we use the fact that $\widetilde{U'_{\mathsf{Ver}}}$ is the same as $\widetilde{U_{\mathsf{Ver}}}$ except that it reads the newly recorded query-answer pairs in $\mathsf{D}_{\mathcal{R}}$ and records them in $\mathsf{D}_{\mathcal{A}}$ and that O only acts on F.

Note that we can also write $|\psi\rangle$ as $\frac{1}{\sqrt{T(n)}}\sum_{t=0}^{T(n)-1} |\psi^{(t)}\rangle |t\rangle_{\mathsf{T}}$ where $|\psi^{(t)}\rangle$ is the state after we run t iterations in the test phase. Then $|\psi^{(t+1)}\rangle = \widetilde{U'_{\mathsf{Ver}}} |\psi^{(t)}\rangle$. As O and $\widetilde{U'_{\mathsf{Ver}}}$ do not run on T, tracing out T does not change the quantity. Therefore,

$$
\begin{aligned}
&\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right]-\Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right]\right|\\
\leq& 6\sqrt{q\left(\mathrm{Tr}(O\left|\psi\right\rangle\left\langle\psi\right|)-\mathrm{Tr}(O\widetilde{U'_{\mathsf{Ver}}}\left|\psi\right\rangle\left\langle\psi\right|\widetilde{U'_{\mathsf{Ver}}}^{\dagger})\right)}\\
=& 6\sqrt{q\left(\frac{1}{T(n)}\sum_{t=0}^{T(n)-1}\mathrm{Tr}\left(O\left|\psi^{(t)}\right\rangle\left\langle\psi^{(t)}\right|\right)-\frac{1}{T(n)}\sum_{t=0}^{T(n)-1}\mathrm{Tr}\left(O\widetilde{U'_{\mathsf{Ver}}}\left|\psi^{(t)}\right\rangle\left\langle\psi^{(t)}\right|\widetilde{U'_{\mathsf{Ver}}}^{\dagger}\right)\right)}\\
=& 6\sqrt{q\left(\frac{1}{T(n)}\sum_{t=0}^{T(n)-1}\mathrm{Tr}\left(O\left|\psi^{(t)}\right\rangle\left\langle\psi^{(t)}\right|\right)-\frac{1}{T(n)}\sum_{t=0}^{T(n)-1}\mathrm{Tr}\left(O\left|\psi^{(t+1)}\right\rangle\left\langle\psi^{(t+1)}\right|\right)\right)}\\
\leq& 6\sqrt{\frac{q}{T(n)}\mathrm{Tr}\left(O\left|\psi^{(0)}\right\rangle\left\langle\psi^{(0)}\right|\right)}\\
\leq& 6\sqrt{\frac{qq'}{T(n)}}
\end{aligned}
$$

where we use the property of compressed oracle techniques that after $q'(n)$ quantum queries, there are at most $q'(n)$ non-$\hat{0}$ elements in F. $\left|\psi^{(0)}\right\rangle$ is just the state we obtain after we run KeyGen and Mint (so there are at most $q'(n)$ quantum queries) and then apply the unitary Comp. So there are at most $q'(n)$ pairs in F of $\left|\psi^{(0)}\right\rangle$. Hence $\mathrm{Tr}\left(O\left|\psi^{(0)}\right\rangle\left\langle\psi^{(0)}\right|\right)\leq q'(n)$.

□

The next claim is an analogue of summing over k of Claim 6 to get a bound for how well $\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}$ behaves on σ_{D_j}. The intuition of the proof is similar to Claim 8.

Claim 9.

$$
\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\ accepts\right]-\Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\ accepts\right]\right|\leq 6\sqrt{\frac{q(n)q'(n)}{N(n)}}
$$

where the probabilities are taken over the randomness of $\mathcal{R}$, the randomness of the inputs to the adversary and the randomness of the adversary (so t and j are both randomized).

Proof. Similar to the proof of Claim 8, these two probabilities can be analyzed by running a sequence of algorithms specificed in Section 6.1 in the compressed view. Let $\left|\phi\right\rangle$ be the same state as in Claim 8 except that now M is the first output register of $\mathcal{A}$ (corresponding to the first synthesized state σ_{D_j}). Let $\left|\psi\right\rangle$ be the state we obtain by running the first step in the compressed view in the case $\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ until the end of the **update** phase of $\mathcal{A}$. Then $\left|\phi\right\rangle$ has the same contents in $\mathsf{D}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{R}}$.

Let $\widetilde{U_{\mathsf{Ver}}}$ and $\widetilde{U'_{\mathsf{Ver}}}$ both run the verification for the state in M, i.e. the synthesized state σ_{D_j}. Recall the fact that $\widetilde{U'_{\mathsf{Ver}}}$ is the same as $\widetilde{U_{\mathsf{Ver}}}$ except that it also records query-answer pairs for $\mathcal{A}$

and that O only acts on F. From Claim 7,

$$\begin{aligned}
&\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\text{ accepts}\right]-\Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\text{ accepts}\right]\right|\\
\leq&6\sqrt{q\left(\mathrm{Tr}(O\left|\phi\right\rangle\left\langle\phi\right|)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\left|\phi\right\rangle\left\langle\phi\right|\widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Syn}}}\left|\psi\right\rangle\left\langle\psi\right|\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U_{\mathsf{Syn}}}^\dagger)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\widetilde{U_{\mathsf{Syn}}}\left|\psi\right\rangle\left\langle\psi\right|\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O\left|\psi\right\rangle\left\langle\psi\right|)-\mathrm{Tr}(O\widetilde{U_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\left|\psi\right\rangle\left\langle\psi\right|\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U_{\mathsf{Ver}}}^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O\left|\psi\right\rangle\left\langle\psi\right|)-\mathrm{Tr}(O\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\left|\psi\right\rangle\left\langle\psi\right|\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)\right)}
\end{aligned}$$

This is because one of $\widetilde{U_{\mathsf{Syn}}}$ is for the second output register, which commutes with $\widetilde{U_{\mathsf{Ver}}}$ (it verifies the state in M). Moreover, both of $\widetilde{U_{\mathsf{Syn}}}$ do not act on F, and thus commute with O.

Note that we can write $\left|\psi\right\rangle=\frac{1}{\sqrt{N(n)}}\sum_{j=0}^{N(n)-1}\left|\psi^{(j)}\right\rangle\left|j\right\rangle_{\mathsf{J}}$ where $\left|\psi^{(j)}\right\rangle$ is the state we obtain after we run a randomized number of iterations in the test phase and j iterations in the update phase. Then $\left|\psi^{(j+1)}\right\rangle=\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\left|\psi^{(j)}\right\rangle$. Notice that O, $\widetilde{U_{\mathsf{Syn}}}$ and $\widetilde{U'_{\mathsf{Ver}}}$ do not run on J. So tracing out J won't change the value. Therefore,

$$\begin{aligned}
&\left|\Pr\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\text{ accepts}\right]-\Pr\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\text{ accepts}\right]\right|\\
\leq&6\sqrt{q\left(\mathrm{Tr}(O\left|\psi\right\rangle\left\langle\psi\right|)-\mathrm{Tr}(O\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\left|\psi\right\rangle\left\langle\psi\right|\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)\right)}\\
=&6\sqrt{q\left(\mathrm{Tr}(O\mathrm{Tr}_{\mathsf{J}}(\left|\psi\right\rangle\left\langle\psi\right|))-\mathrm{Tr}(O\widetilde{U'_{\mathsf{Ver}}}\widetilde{U_{\mathsf{Syn}}}\mathrm{Tr}_{\mathsf{J}}(\left|\psi\right\rangle\left\langle\psi\right|)\widetilde{U_{\mathsf{Syn}}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger)\right)}\\
=&6\sqrt{q\left(\frac{1}{N(n)}\sum_{j=0}^{N(n)-1}\mathrm{Tr}\left(O\left|\psi^{(j)}\right\rangle\left\langle\psi^{(j)}\right|\right)-\frac{1}{N(n)}\sum_{j=0}^{N(n)-1}\mathrm{Tr}\left(O\widetilde{U'_{\mathsf{Ver}}}U_{\mathsf{Syn}}\left|\psi^{(j)}\right\rangle\left\langle\psi^{(j)}\right|U_{\mathsf{Syn}}^\dagger\widetilde{U'_{\mathsf{Ver}}}^\dagger\right)\right)}\\
=&6\sqrt{q\left(\frac{1}{N(n)}\sum_{j=0}^{N(n)-1}\mathrm{Tr}\left(O\left|\psi^{(j)}\right\rangle\left\langle\psi^{(j)}\right|\right)-\frac{1}{N(n)}\sum_{j=0}^{N(n)-1}\mathrm{Tr}\left(O\left|\psi^{(j+1)}\right\rangle\left\langle\psi^{(j+1)}\right|\right)\right)}\\
\leq&6\sqrt{\frac{q}{N(n)}\mathrm{Tr}\left(O\left|\psi^{(0)}\right\rangle\left\langle\psi^{(0)}\right|\right)}\\
\leq&6\sqrt{\frac{qq'}{N(n)}}
\end{aligned}$$

where we use the property of compressed oracle techniques that after $q'(n)$ quantum queries, there are at most $q'(n)$ non-$\hat{0}$ elements in F. $\left|\psi^{(0)}\right\rangle$ is just the state we obtain after we run KeyGen and Mint (so there are at most $q'(n)$ quantum queries) and the test phase of $\mathcal{A}$ (only classical queries) in the compressed view. By the description of $\widetilde{U_R}$, the number of pairs in F can not

increase when we make classical queries. So there are at most $q'(n)$ pairs in F of $|\psi^{(0)}\rangle$. Hence $\mathrm{Tr}\left(O\,|\psi^{(0)}\rangle\langle\psi^{(0)}|\right) \leq q'(n)$. □

Now let's combine the above results to prove Theorem 8.

Proof. Let $\epsilon = 0.01$, $b = 1-\sqrt{1-\delta_r+\epsilon}$, $a = 0.99b$, $T(n) = \frac{36q(n)q'(n)}{\epsilon^2}$ and $N(n) = \frac{q(n)q'(n)}{\epsilon^2(1-\sqrt{1-\delta_r+\epsilon})^4}$.

From Claim 8, $\mathsf{Pr}\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\rho_s^{(t)}))\text{ accepts}\right] \geq \delta_r-\epsilon$. Hence using the same argument in Claim 5, $\mathsf{Pr}\left[\mathsf{Ver}^{D_j,|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\text{ accepts}\right] \geq 0.99(1-\sqrt{1-\delta_r+\epsilon})^2$. From Claim 9, $\mathsf{Pr}\left[\mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle}(pk,(s,\sigma_{D_j}))\text{ accepts}\right] \geq 0.99(1-\sqrt{1-\delta_r+\epsilon})^2-6\sqrt{\frac{q(n)q'(n)}{N(n)}} \geq 0.9(1-\sqrt{1-\delta_r+\epsilon})^2$. Thus by union bound, the outputs of our adversary pass two verifications simultaneously with probability at least $1.8(1-\sqrt{1-\delta_r+\epsilon})^2-1$, which is non-negligible when $\delta_r \geq 0.99$. □

That is, the adversary we construct gives a valid attack to $(\mathsf{KeyGen}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}, \mathsf{Mint}^{|\mathcal{R}\rangle,|\mathsf{PSPACE}\rangle}, \mathsf{Ver}^{\mathcal{R},|\mathsf{PSPACE}\rangle})$ where $\delta_r \geq 0.99$ and $\delta_s = \mathsf{negl}(n)$, which establishes Theorem 7.

References

[AA09] Scott Aaronson and Andris Ambainis. "The need for structure in quantum speedups". In: *arXiv preprint arXiv:0911.0996* (2009) (cit. on p. 3).

[Aar09] Scott Aaronson. "Quantum copy-protection and quantum money". In: *24th Annual IEEE Conference on Computational Complexity*. IEEE Computer Soc., Los Alamitos, CA, 2009, pp. 229–242. DOI: 10.1109/CCC.2009.42. URL: https://doi.org/10.1109/CCC.2009.42 (cit. on p. 1).

[AC13] Scott Aaronson and Paul Christiano. "Quantum money from hidden subspaces". In: *Theory Comput.* 9 (2013), pp. 349–401. DOI: 10.4086/toc.2013.v009a009. URL: https://doi.org/10.4086/toc.2013.v009a009 (cit. on pp. 2, 12).

[ACC+22] Per Austrin, Hao Chung, Kai-Min Chung, Shiuan Fu, Yao-Ting Lin, and Mohammad Mahmoody. "On the Impossibility of Key Agreements from Quantum Random Oracles". In: *Cryptology ePrint Archive* (2022) (cit. on pp. 3, 12).

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. "Indistinguishability obfuscation without multilinear maps: new paradigms via low degree weak pseudorandomness and security amplification". In: *Annual International Cryptology Conference*. Springer. 2019, pp. 284–332 (cit. on p. 2).

[AK22] Prabhanjan Ananth and Fatih Kaleoglu. *A Note on Copy-Protection from Random Oracles*. Cryptology ePrint Archive, Paper 2022/1109. https://eprint.iacr.org/2022/1109. 2022. URL: https://eprint.iacr.org/2022/1109 (cit. on pp. 5, 8, 9, 25).

[AKL+22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. "On the feasibility of unclonable encryption, and more". In: *Annual International Cryptology Conference*. Springer. 2022, pp. 212–241 (cit. on p. 4).

[AL21] Prabhanjan Ananth and Rolando L La Placa. "Secure Software Leasing". In: *Eurocrypt* (2021) (cit. on p. 1).

[BDG22] Andriyan Bilyk, Javad Doliskani, and Zhiyong Gong. "Cryptanalysis of Three Quantum Money Schemes". In: *arXiv preprint arXiv:2205.10488* (2022) (cit. on pp. 2, 12).

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. "Factoring and pairings are not necessary for iO: circular-secure LWE suffices". In: *Cryptology ePrint Archive* (2020) (cit. on p. 2).

[BDV17] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. "Structure vs. hardness through the obfuscation lens". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 696–723 (cit. on p. 2).

[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*. Springer. 2001, pp. 1–18 (cit. on p. 12).

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. "Quantum one-time programs". In: *Annual Cryptology Conference*. Springer. 2013, pp. 344–360 (cit. on p. 1).

[BI20] Anne Broadbent and Rabib Islam. "Quantum encryption with certified deletion". In: *Theory of Cryptography Conference*. Springer. 2020, pp. 92–122 (cit. on p. 1).

[BL20] Anne Broadbent and Sébastien Lord. "Uncloneable Quantum Encryption via Oracles". In: *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*. Ed. by Steven T. Flammia. Vol. 158. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 4:1–4:22. DOI: 10.4230/LIPIcs.TQC.2020.4 (cit. on p. 1).

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. "Merkle puzzles are optimal—an o (n 2)-query attack on any key exchange from a random oracle". In: *Annual International Cryptology Conference*. Springer. 2009, pp. 374–390 (cit. on p. 2).

[BPR+08] Dan Boneh, Periklis Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. "On the impossibility of basing identity based encryption on trapdoor permutations". In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2008, pp. 283–292 (cit. on p. 2).

[BS16] Shalev Ben-David and Or Sattath. "Quantum tokens for digital signatures". In: *arXiv preprint arXiv:1609.09047* (2016) (cit. on p. 1).

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. "On obfuscation with random oracles". In: *Theory of Cryptography Conference*. Springer. 2015, pp. 456–467 (cit. on pp. 8, 25).

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. "Hidden cosets and applications to unclonable cryptography". In: *Annual International Cryptology Conference*. Springer. 2021, pp. 556–584 (cit. on pp. 1, 4).

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. "Quantum copy-protection of compute-and-compare programs in the quantum random oracle model". In: *arXiv preprint arXiv:2009.13865* (2020) (cit. on p. 4).

[CMSZ22] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. "Post-quantum succinct arguments: breaking the quantum rewinding barrier". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science—FOCS 2021*. IEEE Computer Soc., Los Alamitos, CA, [2022] ©2022, pp. 49–58. DOI: 10.1109/FOCS52979.2021.00014. URL: https://doi.org/10.1109/FOCS52979.2021.00014 (cit. on pp. 16, 17).

[CX21] Shujiao Cao and Rui Xue. "Being a permutation is also orthogonal to one-wayness in quantum world: Impossibilities of quantum one-way permutations from one-wayness primitives". In: *Theoretical Computer Science* 855 (2021), pp. 16–42 (cit. on p. 12).

[DG17] Nico Döttling and Sanjam Garg. "Identity-based encryption from the Diffie-Hellman assumption". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 537–569 (cit. on p. 2).

[Die82] DGBJ Dieks. "Communication by EPR devices". In: *Physics Letters A* 92.6 (1982), pp. 271–272 (cit. on pp. 1, 3).

[DLMM11] Dana Dachman-Soled, Yehuda Lindell, Mohammad Mahmoody, and Tal Malkin. "On the black-box complexity of optimally-fair coin tossing". In: *Theory of Cryptography Conference*. Springer. 2011, pp. 450–467 (cit. on p. 2).

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. "Succinct LWE sampling, random polynomials, and obfuscation". In: *Theory of Cryptography Conference*. Springer. 2021, pp. 256–287 (cit. on p. 2).

[FGH+12] Edward Farhi, David Gosset, Avinatan Hassidim, Andrew Lutomirski, and Peter Shor. "Quantum money from knots". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2012, pp. 276–289 (cit. on pp. 2, 12).

[GKLM12] Vipul Goyal, Virendra Kumar, Satya Lokam, and Mohammad Mahmoody. "On black-box reductions between predicate encryption schemes". In: *Theory of Cryptography Conference*. Springer. 2012, pp. 440–457 (cit. on p. 2).

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. "The relationship between public key encryption and oblivious transfer". In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE. 2000, pp. 325–335 (cit. on p. 2).

[GP21] Romain Gay and Rafael Pass. "Indistinguishability obfuscation from circular security". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 736–749 (cit. on p. 2).

[GZ20] Marios Georgiou and Mark Zhandry. "Unclonable decryption keys". In: *Cryptology ePrint Archive* (2020) (cit. on p. 1).

[HJL21] Sam Hopkins, Aayush Jain, and Huijia Lin. "Counterexamples to new circular security assumptions underlying iO". In: *Annual International Cryptology Conference*. Springer. 2021, pp. 673–700 (cit. on p. 2).

[HY20] Akinori Hosoyamada and Takashi Yamakawa. "Finding collisions in a quantum world: quantum black-box separation of collision-resistance and one-wayness". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 3–32 (cit. on p. 12).

[IR90] Russell Impagliazzo and Steven Rudich. "Limits on the provable consequences of one-way permutations". In: *Advances in cryptology—CRYPTO '88 (Santa Barbara, CA, 1988)*. Vol. 403. Lecture Notes in Comput. Sci. Springer, Berlin, 1990, pp. 8–26. DOI: 10.1007/0-387-34799-2_2. URL: https://doi.org/10.1007/0-387-34799-2_2 (cit. on p. 3).

[JLS18] Zhengfeng Ji, Yi-Kai Liu, and Fang Song. "Pseudorandom quantum states". In: *Annual International Cryptology Conference*. Springer. 2018, pp. 126–152 (cit. on p. 12).

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability obfuscation from well-founded assumptions". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 60–73 (cit. on p. 2).

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability Obfuscation from LPN over, DLIN, and PRGs in NC". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022, pp. 670–699 (cit. on p. 2).

[KLS22] Andrey Boris Khesin, Jonathan Z Lu, and Peter W Shor. "Publicly verifiable quantum money from random lattices". In: *arXiv preprint arXiv:2207.13135* (2022) (cit. on pp. 2, 12).

[KSS21] Daniel M Kane, Shahed Sharif, and Alice Silverberg. "Quantum money from quaternion algebras". In: *arXiv preprint arXiv:2109.12643* (2021) (cit. on pp. 2, 12).

[Lut10] Andrew Lutomirski. "An online attack against Wiesner's quantum money". In: *arXiv preprint arXiv:1010.0256* (2010) (cit. on p. 12).

[MVW12] Abel Molina, Thomas Vidick, and John Watrous. "Optimal counterfeiting attacks and generalizations for Wiesner's quantum money". In: *Conference on Quantum Computation, Communication, and Cryptography*. Springer. 2012, pp. 45–64 (cit. on p. 12).

[MW05] Chris Marriott and John Watrous. "Quantum arthur–merlin games". In: *computational complexity* 14.2 (2005), pp. 122–152 (cit. on pp. 6, 16, 17, 20).

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, 2000, pp. xxvi+676. ISBN: 0-521-63235-8; 0-521-63503-9 (cit. on p. 13).

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994 (cit. on p. 14).

[PRV12] Periklis A Papakonstantinou, Charles W Rackoff, and Yevgeniy Vahlis. "How powerful are the DDH hard groups?" In: *Cryptology ePrint Archive* (2012) (cit. on p. 2).

[Rob21] Bhaskar Roberts. "Security analysis of quantum lightning". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2021, pp. 562–567 (cit. on pp. 2, 12).

[RS22] Roy Radian and Or Sattath. "Semi-quantum money". In: *Journal of Cryptology* 35.2 (2022), pp. 1–70 (cit. on p. 12).

[RTV04] Omer Reingold, Luca Trevisan, and Salil Vadhan. "Notions of reducibility between cryptographic primitives". In: *Theory of Cryptography Conference*. Springer. 2004, pp. 1–20 (cit. on p. 2).

[Rud91] Steven Rudich. "The Use of Interaction in Public Cryptosystems." In: *Annual International Cryptology Conference*. Springer. 1991, pp. 242–251 (cit. on p. 2).

[RY21] Gregory Rosenthal and Henry Yuen. "Interactive Proofs for Synthesizing Quantum States and Unitaries". In: *arXiv preprint arXiv:2108.07192* (2021) (cit. on pp. 6, 7, 15).

[Shm22a] Omri Shmueli. "Public-key Quantum money with a classical bank". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 790–803 (cit. on pp. 2, 12).

[Shm22b] Omri Shmueli. "Semi-Quantum Tokenized Signatures". In: *Cryptology ePrint Archive* (2022) (cit. on p. 2).

[Sim98] Daniel R Simon. "Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?" In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1998, pp. 334–345 (cit. on pp. 2, 12).

[Unr16] Dominique Unruh. "Computationally binding quantum commitments". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2016, pp. 497–527 (cit. on p. 4).

[Wie83] Stephen Wiesner. "Conjugate coding". In: *ACM Sigact News* 15.1 (1983), pp. 78–88 (cit. on pp. 1, 12).

[Win99] Andreas Winter. "Coding theorem and strong converse for quantum channels". In: *IEEE Transactions on Information Theory* 45.7 (1999), pp. 2481–2485 (cit. on pp. 5, 16).

[WW21] Hoeteck Wee and Daniel Wichs. "Candidate obfuscation via oblivious LWE sampling". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2021, pp. 127–156 (cit. on p. 2).

[WZ82] William K Wootters and Wojciech H Zurek. "A single quantum cannot be cloned". In: *Nature* 299.5886 (1982), pp. 802–803 (cit. on pp. 1, 3).

[Zha15] Mark Zhandry. "A note on the quantum collision and set equality problems". In: *Quantum Information & Computation* 15.7-8 (2015), pp. 557–567 (cit. on p. 4).

[Zha18] Mark Zhandry. *How to Record Quantum Queries, and Applications to Quantum Indifferentiability*. Cryptology ePrint Archive, Paper 2018/276. https://eprint.iacr.org/2018/276. 2018. URL: https://eprint.iacr.org/2018/276 (cit. on pp. 11, 17, 32).

[Zha21] Mark Zhandry. "Quantum lightning never strikes the same state twice. Or: quantum money from cryptographic assumptions". In: *J. Cryptology* 34.1 (2021), Paper No. 6, 56. ISSN: 0933-2790. DOI: 10.1007/s00145-020-09372-x. URL: https://doi.org/10.1007/s00145-020-09372-x (cit. on pp. 2, 4, 5, 12).

A Appendix

Table 1: Parameters in Section 6.3

$q(n), q'(n)$	n is the security number. Ver makes $q(n)$ classical queries. KeyGen and Mint make $q'(n)$ quantum queries in total. We sometimes omit n.
$T(n), N(n)$	two polynomials that will decide the maximal possible number of iterations we run in the test phase and the update phase.

Table 2: Registers in Section 6.3

Pk	the register storing the public key
S	the register storing the serial number
M	the register storing the alleged money state
$\mathsf{D}_{\mathcal{A}}$	the register storing the classical queries database maintained by $\mathcal{A}$
$\mathsf{D}_{\mathcal{R}}$	the register storing the classical queries we made so far along with their answers (maintained by the oracle)
F	the register storing the oracle if in the decompressed view or the register storing D_F (the database for non-$\|\hat{0}\rangle$ elements) if in compressed view
G, H	the register storing unimportant things for the analysis. For example, it may include the secret key, working space for KeyGen and Mint, and some unused fresh ancillas.
T	the register storing the number of iterations for test phase.
J	the register storing the number of iterations for update phase.
$\mathsf{Q}, \mathsf{Q}_1, \mathsf{Q}_2$	the register storing the next query position.
$\mathsf{A}, \mathsf{A}_1, \mathsf{A}_2$	the register to store the next query answer.

Table 3: States in Section 6.3

$\|\phi\rangle$	A state in the following form (i.e. it's in the compressed view and the contents in $\mathsf{D}_{\mathcal{R}}$ and $\mathsf{D}_{\mathcal{A}}$ are the same), $$\|\phi\rangle = \sum_{\substack{pk,s,m,D,D_F,g \\ \text{s.t. } D\cap D_F=\emptyset}} \alpha_{pk,s,m,D,D_F,g} \|pk\rangle_{\mathsf{Pk}} \|s\rangle_{\mathsf{S}} \|m\rangle_{\mathsf{M}} \|D\rangle_{\mathsf{D}_{\mathcal{A}}} \|D_F\rangle_{\mathsf{F}} \|D\rangle_{\mathsf{D}_{\mathcal{R}}} \|g\rangle_{\mathsf{G}}.$$ In Claim 8 and Claim 9, we will instantiate it with the pure state we obtain by applying the unitaries $\widetilde{U_{\mathsf{KeyGen}}}$, $\widetilde{U_{\mathsf{Mint}}}$ and $\widetilde{U_{\mathcal{A}}}$ to the state $\|1^n\rangle \|\emptyset\rangle_{\mathsf{D}_{\mathcal{A}}} \|\emptyset\rangle_{\mathsf{F}} \|\emptyset\rangle_{\mathsf{D}_{\mathcal{R}}}$ along with enough ancillas.
$\|\phi_j\rangle$	$\|\phi_j\rangle = \widetilde{U_C}U_{j-1}\cdots\widetilde{U_C}U_0\|\phi\rangle$. It's the state when we run $\mathsf{Ver}^{\mathcal{R},\|\mathsf{PSPACE}\rangle}$ on $\|\phi\rangle$ in compressed view until we have answered the j^{th} query.
$\|\psi\rangle$	We abuse the notation. $\|\psi\rangle$ is the pure state we obtain by running the first step in the compressed view in the case $\mathsf{Ver}^{\mathcal{R},\|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ until the end of the **test** phase (in Claim 8) or the **update** phase (in Claim 9) of $\mathcal{A}$.
$\|\varphi\rangle$	an arbitary state ready for two "parallel" classical queries on different registers.
$\|\psi_0\rangle$	the pure state we obtain by running the first step in the compressed view in the case $\mathsf{Ver}^{\mathcal{R},\|\mathsf{PSPACE}\rangle}(pk,(s,\cdot))$ on $\rho_s^{(t)}$ until we finish the **test** phase and then truncating J.
$\|\psi^{(t)}\rangle, \|\psi^{(j)}\rangle$	We abuse the notation. In Claim 8, $\|\psi^{(t)}\rangle$ is the state after we run t iterations in the test phase but not run the update phase in the compressed view. In Claim 9, $\|\psi^{(j)}\rangle$ is the state we obtain after we run a randomized number of iterations in the test phase and j iterations in the update phase in the compressed view.

Table 4: Observable and Unitaries in Section 6.3

O	the observable corresponding to the number of pairs in F (i.e. half of the nonempty length in F). Formally, $O = \sum_{D_F} \lvert D_F\rvert \lvert D_F\rangle\langle D_F\rvert_{\mathsf{F}}$ where $\lvert D_F\rvert$ is the number of pairs in D_F. It will only be applied to those states in compressed view.
Decomp	the unitary defined in Section 6.2. It acts on $\mathsf{D}_{\mathcal{R}}$ and F and decompresses two databases to one database and the oracle.
Comp	the inverse of the unitary Decomp.
$\widetilde{U}$	$\widetilde{U} = \mathsf{Comp}U\mathsf{Decomp}$. It's the compressed view version of U for a general unitary U. See the figure in Section 6.2 for more details.
U_Q	the unitary corresponding to answering a **quantum** query with the real oracle.
U_C	the unitary corresponding to answering a **classical** query with the real oracle.
U_R	the same as U_C except that it records the query-answer for $\mathcal{A}$ at the same time.
U_D	the unitary corresponding to answering a classical query with the database in register $\mathsf{D}_{\mathcal{A}}$ while recording the query-answer to $\mathsf{D}_{\mathcal{A}}$ for later use.
U'_D	the unitary corresponding to answering a classical query with the database in register $\mathsf{D}_{\mathcal{R}}$ while recording the query-answer to $\mathsf{D}_{\mathcal{R}}$ for later use.
U_i	When $0 \le i \le q-1$, it's the unitary corresponding to the preparation of the $(i+1)^{th}$ query of Ver. When $i = q$, it's the unitary after the final query of Ver.
$U_{\mathsf{KeyGen}}, U_{\mathsf{Mint}}$	the unitary $U_{\mathsf{KeyGen},n}, U_{\mathsf{Mint},n}$ defined in Section 6.1.
$U_{\mathsf{Ver}}, U_{\mathsf{Syn}}$	the unitary $U_{\mathsf{Ver},n}, U_{\mathsf{Syn},n}$ defined in Section 6.1, $U_{\mathsf{Ver}} = U_q U_C U_{q-1} \cdots U_C U_0$.
U'_{Ver}	the unitary corresponding to doing the verification while recording the query-answer pair for $\mathcal{A}$, $U'_{\mathsf{Ver}} = U_q U_R U_{q-1} \cdots U_R U_0$.
U_{Sim}	the unitary corresponding to running $\mathsf{Ver}^{D,\lvert\mathsf{PSPACE}\rangle}$ where D is the content in $\mathsf{D}_{\mathcal{A}}$, $U_{\mathsf{Sim}} = U_q U_D U_{q-1} \cdots U_D U_0$.
U_{Upd}	the unitary defined in Section 6.1 that describes our update phase. Formally, it's the unitary $\sum_{j=0}^{N(n)-1} (U_{\mathsf{Ver}} U_{\mathsf{Syn}})^j \lvert j\rangle\langle j\rvert_{\mathsf{J}}$.