Quantum Immune One-Time Memories

Qipeng Liu[1], Amit Sahai[2], Mark Zhandry[1]

[1] Princeton University & NTT Research
[2] University of California, Los Angeles

Abstract. One-time memories (OTM) are the hardware version of oblivious transfer, and are useful for constructing objects that are impossible with software alone, such as one-time programs. In this work, we consider attacks on OTMs where a quantum adversary can leverage his physical access to the memory to mount quantum "superposition attacks" against the memory. Such attacks result in significantly weakened OTMs. For example, in the application to one-time programs, it may appear that such an adversary can always "quantumize" the classical protocol by running it on a superposition of inputs, and therefore learn superpositions of outputs of the protocol.

Perhaps surprisingly, we show that this intuition is false: we construct one-time programs from quantum-accessible one-time memories where the view of an adversary, despite making quantum queries, can be simulated by making only classical queries to the ideal functionality. At the heart of our work is a method of immunizing one-time memories against superposition attacks.

1 Introduction

Recent demonstrations have confirmed the enhanced computational power of quantum computers [Aru19], and it is widely believed that full-scale quantum computers will eventually be viable. In this case, much of the cryptography used today will be broken [Sho94], meaning traditional cryptosystems will need to be replaced by quantum-immune systems, such as those based on lattices.

Recently, a different kind of quantum threat has been identified: superposition attacks [KM10, Zha12, BZ13a, BZ13b, DFNS14, KLLN16, Zha16, GHS16, ATTU16, GYZ17, SS17]. In these attacks, the adversary uses quantum "superpositions" of messages to interrogate the honest users of the system. Various primitives have been considered in this setting, including PRFs, MACs, signatures, and encryption.

In these works on superposition attacks, the motivation is typically that the cryptosystem is being implemented on a piece of secure hardware such as a smart card. The adversary is then able to temporarily access the hardware, and use it's physical access to interrogate the hardware using quantum messages. Given that there is currently no full-scale quantum computer to experiment with, it is difficult to predict what kinds of attacks will be possible in such settings. As such, it is important to conservatively model security, in case such quantum attacks are possible.

For example, even classical processors exhibit quantum phenomena due to their small scale; perhaps a quantum attacker with physical access to such a device can leverage the laws of quantum mechanics to extract information from the device. Worse yet, once quantum computers become commonplace and cryptosystems are actually being run on quantum computers, it may be possible to directly query the system on quantum messages. The goal is then to maintain security even in the presence of such attacks.

Tamper-proof Hardware. Despite the hardware setting being the underlying motivation for these various security models, the literature has not yet actually considered superposition attacks on tamper-proof hardware tokens. Tamper proof hardware tokens allow whomever possesses the token to query the token as a black box; the hardware assumption is that any stronger access is impossible.

There have been a number of works considering the hardware token model classically [Kat07, CGS08, MS08, GKR08, GIS+10]. The goal is typically to start with a very basic hardware token, and build a much more complex object, usually one that is impossible using just software alone. One of the most commonly considered hardware tokens is a *one-time memory* (OTM), which contains two input bits x_0, x_1. The token allows for just a single query on a bit b, to which it responds with x_b, and afterward "self destructs" and refuses additional queries. One-time memories, for example, are known to be sufficient to build one-time programs [GKR08, GIS+10], where whomever posses the program can query it on a single input x to learn a single output $f(x)$, but nothing more.

There have been relatively few works on tamper proof hardware tokens in the quantum setting. It has been shown that classical one-time memories are sufficient to build *quantum* one-time programs [BGS13], where the input x is replaced by a quantum state, and the function f is a general quantum circuit. More recently, it has been shown how to achieve quantum one-time programs from *stateless* classical hardware tokens [BGZ18].

However, so far these works in the quantum setting have all considered the hardware token as accepting only *classical* queries. Yet, in the hardware token model, the adversary is *inherently* given complete physical access to the token. If the adversary has a quantum computer, he can thus attempt to interrogate the token in superposition. One may design new hardware tokens that will try to "classicalize" incoming messages by measuring them, defeating such attacks. However, this requires a new and very strong hardware assumption: namely that the token can effectively perform the needed measurement, despite the adversary's ability to physically manipulate the token. This assumption may be unreasonable in a variety of restricted hardware settings. For example, according to the many-worlds interpretation of quantum mechanics, such measurements amount to entangling the query with another system. Typically, the other system is the environment; but in the hardware setting the adversary controls the entire environment, which could in principle allow him to undo the measurement. Alternatively, the other system could be secured inside the hardware token itself. However, in this case, if this other system de-coheres — which amounts to entangling with the adversarially-controlled environment — then the adversary can in

principle similarly undo the measurement. Even in ideal conditions, maintaining coherence is extremely difficult; in our setting the token is under the control of the adversary, who can subject the token to challenging conditions in order cause de-coherence.

Upgrading OTMs for quantum-access security. It is therefore a more conservative — and arguably more reasonable — hardware assumption to allow the token to accept quantum queries from the adversary. On the other hand, this "quantum accessible" hardware token model seems very limiting. For example, in the case of one-time memories, one can run the Deutsch-Jozsa algorithm [DJ92] to learn the parity of the two input bits, something that is not possible classically. Plugging such quantum accessible one-time memories into existing results will thus completely invalidate their proof.

Worse yet, consider implementing a supposed one-time program using quantum-accessible one-time memories. An adversary that can query the one-time memories on quantum superpositions can always run one-time program honestly, but run it on a superposition of inputs. More precisely, the adversary initially prepares a superposition $\sum_{b,c} \alpha_{b,c}|b, c, 0\rangle$ where b are inputs, c are responses, and the 0 is a set of registers initialized to 0 that will serve as workspace. The adversary then runs the evaluation of the one-time program in superposition using the input b to generate its superpositions that it sends to the various one-time memories, and XORing the result of its computation into the response register c. It would appear that the adversary is simulating a *quantum* oracle query to the functionality. Using such quantum access, it could then run some single-query quantum algorithm, such as quantum period finding, to learn information that was not possible classically.

In this work, we initiate the study of classical tamper-proof hardware tokens, in the setting where the adversary can potentially gain superposition access to the token. As discussed above, this is the natural and conservative way to model such tokens in the quantum setting. Given the above discussion, it is also unclear a priori if any of the classical feasibility results can be re-created in this setting. For this initial work in the area, we focus in particular on the following natural question:

What is the best security we can achieve for one-time programs using quantum-accessible one-time memories?

We stress that our goal is to understand what security is possible from *classical* OTM tokens when subjected to quantum-access attacks; we are not interested in designing new hardware tokens from scratch for the purpose of emulating OTMs, which would require new hardware assumptions.

1.1 Our Contributions

In this paper, we construct secure one-time programs in the form of one-time memories, which can be evaluated by classical parties, but maintain security

even if the adversary is quantum and can make quantum superposition queries to the one-time memories. More precisely, even though the adversary has quantum access to the underlying one-time memories, we give a scheme where the adversary can only learn a single *classical* output of the protocol.

Given the discussion above, this seems like a contradiction: what if the adversary runs a single-query quantum period finding algorithm by running the entire program in superposition? Our main insight is to show, perhaps surprisingly, that the "run honest protocol in superposition" attack sketched above can actually be made to fail.

To see why this might at all be possible, consider the hypothetical quantum superposition attacker from above. Our key observation is that, in running the protocol, a significant amount of intermediate values may have been generated and stored to the adversary's work space. To mimic a quantum query to the program, the adversary needs to create the state $\sum_{b,c} \alpha_{b,c}|b, c \oplus f(b)\rangle$. But at the end of the protocol the adversary instead has the state $\sum_{b,c} \alpha_{b,c}|b, c \oplus f(b), w(b)\rangle$ where $w(b)$ are the intermediate values generated on input b that are used to compute $f(b)$. In the classical setting, the adversary can just throw away $w(b)$. However, in the quantum setting, throwing away $w(b)$ is equivalent to measuring $w(b)$, which will destroy the adversary's superposition. Instead, $w(b)$ must actually be *un-computed* by running the computation a second time. The catch is that the adversary will therefore need to run the whole protocol twice, but the one-time memories will not allow that to happen. Therefore, if the protocol requires storing many intermediate values, it may be impossible for the adversary to actually effectively query the program in superposition.

The main technical hurdle is then to design a scheme and analysis where the generated intermediate values *provably* cause an effective measurement on the input registers b.

Our main technical result is to show how to immunize one-time memories against quantum access. That is, we show how to *use several quantum-accessible one-time memories* to construct a one-time memory that is immune to superposition attacks. Even though the adversary has the capability to interrogate the constituent memories in superposition, we show that our construction can essentially only be queried on a single classical input. This is formalized by requiring that the view of the adversary can be (efficiently) simulated by a simulator that is granted only a single *classical* query to the ideal one-time memory functionality. This transformation is information-theoretic, relying on no computational assumptions. Our new classical-access one-time memories can then be plugged into existing information-theoretic constructions [GIS+10] to achieve one-time programs that can only be queried on a single classical input, despite the underlying one-time memories accepting quantum queries. We can also plug into the results of [BGS13] to achieve secure one-time *quantum* programs.

1.2 A First Step: The Linear Access Model

In order to motivate our scheme, we consider the following. A quantum-accessible one-time memory is a hardware version of a familiar setting, where a single quan-

tum query is made to a two-bit string. It is known that it is impossible to learn both input bits with certainty (see, for example, [BZ13a] for a proof of a much more general statement). On the other hand, by applying the Deutsch-Jozsa algorithm [DJ92], an adversary can learn the parity of the two bits, something that is impossible classically.

Suppose, then, that the adversary is limited to either making a classical query, or to applying the Deutsch-Jozsa algorithm. This means it can learn any single linear function on the input bits. Therefore, a simpler purely classical setting can model such an adversary: namely, the one-time memory interface now lets the adversary learn any single linear function of its choice. This simpler classical setting will be the starting point for our quantum analysis. Then as a starting point, we ask: how can we immunize such weak "linearly-queriable" one-time memories, to achieve a one-time memory scheme meeting the standard classical notion of security?

Guided by this linear-access model, we design a simple construction to achieve the desired immunization. Let z_0, z_1 be the input bits. We first observe the following: suppose the procedure to reconstruct z_b from the output of the linear-access one-time memories is itself a linear operation. Then one can actually learn any linear function of the z_b by making appropriate linear queries to the underlying OTMs. Even more, in the full quantum-access setting, the adversary can query the z's in superposition, *without requiring any work space computations*, meaning the adversary can actually correctly simulate a quantum query to the z's. Thus, a scheme with linear reconstruction does not provide any improved security.

To introduce the needed non-linearity, we construct the following simple and natural scheme. We will choose four random vectors $\mathbf{x}_0, \mathbf{y}_0, \mathbf{x}_1, \mathbf{y}_1 \in \{0,1\}^\lambda$ conditioned on $\mathbf{x}_0 \cdot \mathbf{y}_0 = z_0, \mathbf{x}_1 \cdot \mathbf{y}_1 = z_1$. We will then use 2λ quantum-accessible one-time memories, λ OTMs with inputs $(x_{0,i}, x_{1,i})$ and another λ with inputs $(y_{0,i}, y_{1,i})$. An honest party, on input b, will simply query each of the quantum accessible OTMs on b to get the vectors $\mathbf{x}_b, \mathbf{y}_b$. It then takes the inner product of the results to learn z_b.

On the other hand, a malicious attacker can submit much more complicated queries, mixing different b for each quantum accessible OTM, and even running each OTM on other linear combinations. Suppose for example the receiver tries to learn $z_0 \oplus z_1$, which can be computed as the XOR of the values $x_{0,i}y_{0,i} + x_{1,i}y_{1,i}$ for each i. Suppose the adversary queries the OTM containing $(x_{0,i}, x_{1,i})$ first. The adversary then needs to learn $x_{0,i}y_{0,i} + x_{1,i}y_{1,i}$. The adversary queries for a linear combination $s_{0,i}x_{0,i} + s_{1,i}x_{1,i}$. If the adversary happened to know both $y_{0,i}, y_{1,i}$, it could set $s_{b,i} = y_{b,i}$ and it would be done. However, at this point, the adversary knows nothing about $(y_{0,i}, y_{1,i})$, and must commit to querying a linear combination before seeing these values. If it committed to a linear combination $(s_{0,i}, s_{1,i})$ that was incorrect, it has no way to later change its mind and learn

the right linear combination[3]. This is true, even if we later give the adversary *both* values of y.

By symmetry, the same thing happens if the adversary queries for the ith component of y before x. Therefore, in either case there is some probability that it learns the incorrect linear combination. Over all queries, the probability it guesses the correct linear combination for all queries is exponentially small.

While the above intuition suggests a classical proof strategy, we need to actually prove the full quantum security of our protocol. Now, the adversary can perform even more complex operations, resulting in not just classical information, but also quantum states that may depend on both inputs in ways that cannot be explained by a simple linear combination of the inputs. Moreover, it is not enough to show that the adversary's view is independent of one of the z_b: in order to prove security, we actually need to show how to *simulate* such an adversary given just a single *classical* query to z_0 or z_1. Guided by the previous discussions, we would hope that the work space produced during the protocol execution would allow the simulator to measure the adversary's queries to learn the classical bit b, which it then queries to learn z_b. However, the adversary could have arbitrarily deviated from the correct protocol in a way that produced no work space bits at all. In this case, any attempt to measure the adversary would be detected, leading to an incorrect simulation. It is therefore not clear a priori whether such simulation should even be possible.

1.3 Analyzing Density Matrices

Towards our solution, observe that the adversary's state after making the quantum-accessible OTM queries is a *mixed* state: for any choice of randomness in the construction, the receiver gets a quantum state, and its overall state is in some sense the average of these randomness-dependent states. Such states are characterized by a density matrix. The density matrix has the form $\sum \rho_{a,a'}|a\rangle\langle a'|$, where $|a\rangle$ represents column vectors and $\langle a'|$ represents row vectors. The density matrix for a pure state $|\psi\rangle$ is simply $|\psi\rangle\langle\psi|$, and the density matrix for a mixture of two other states is simply the component-wise average.

By analyzing this density matrix, we show that there is an equivalent way to generate it in two steps: (1) perform a partial measurement on the adversary's query state that is *independent* of z_0, z_1, and (2) perform additional operations depending on z_0, z_1. Step (1) comes from the fact that the adversary has generated many intermediate computations — namely the results of all the quantum accessible OTM queries — that it cannot easily un-compute. Remarkably, by performing the partial measurement in step (1), we show that we actually have enough information to simulate step (2), while only needing to make a single *classical* query to z_0 or z_1.

In more detail, we note that for "early" queries, where, say, at most half of the x memories and at most half of the y memories have been queried, the values

[3] Note that an adversary can always later pretend it queried on the combination $0 \times x_{0,i} + 0 \times x_{1,i}$ after the fact by throwing out its response and replacing it with 0. However, no other changes are possible.

of x and y are statistically close to random and independent of z_0, z_1. Therefore, we can imagine simulating the value for x and y on-the-fly, choosing them at random for the early queries. At some point, we stop the on-the-fly simulation, and sample all remaining values from the appropriate conditional distribution.

Suppose the adversary is making an early query to the OTM containing $(x_{0,i}, x_{1,i})$, and suppose that the $(y_{0,i}, y_{1,i})$ memory has not been queried. We note consider density matrix ρ for the joint state of the adversary and its query immediately before the query is processed. For simplicity in this discussion, we will treat the adversary as having no state outside of its query, but it is straightforward to generalize to that includes a non-empty state.

The adversaries query to the OTM is thus two qubits, which we will call $|b\rangle$ (indicating which value it is interested in) and $|c\rangle$ (where the response will be recorded). We will now analyze the effect of the query in *computational* basis for $|b\rangle$ and the *Hadamard* basis for $|c\rangle$. This means the query maps $|b, c\rangle$ to $(-1)^{cx_{b,i}}|b, c\rangle$. Examining ρ in the same basis, suppose ρ starts off as the state $\sum \rho_{b,c,b',c'}|b, c\rangle\langle b', c'|$. Then after the query the density matrix becomes $\rho' = \sum \rho_{b,c,b',c'}|b, c\rangle\langle b', c'|(-1)^{cx_{b,i}+c'x_{b',i}}$.

Now, consider the case $y_{0,i} = 0$. This means that in order to compute z_0, the value $x_{0,i}$ is not needed. In other words, $x_{0,i}$ remains random and independent of z_0, even after conditioning on the remaing values that have yet to be queried. This means we can "trace" out the $x_{0,i}$ in the adversary's state. In other words, we compute the mixture of the two versions of this state, corresponding to $x_{0,i} = 0, 1$. To carry out the tracing, we write $x_{b,i} = (1-b)x_{0,i} + bx_{1,i}$. Then the phase component becomes

$$(-1)^{cx_{b,i}+c'x_{b',i}} = (-1)^{(c(1-b)+c'(1-b'))x_{0,i}+(cb+c'b')x_{1,i}}$$

If we carry out the trace, we simply average these terms for $x_{0,i} = 0, 1$. Terms with $c(1-b) \neq c'(1-b')$ will average to zero, and terms with $c(1-b) = c'(1-b')$ will average to $(-1)^{(cb+c'b')x_{1,i}}$. The result is the density matrix after the query and tracing out $x_{0,i}$ is equivalent to the following:

- First, zero out terms with $c(1-b) \neq c'(1-b')$
- Second, multiply by $(-1)^{(cb+c'b')x_{1,i}}$

We now observe that zeroing out terms with $c(1-b) \neq c'(1-b')$ has the exact same effect as *measuring* $c(1-b)$. Moreover, applying the phase $(-1)^{(cb+c'b')x_{1,i}}$ to a density matrix is equivalent to applying the phase $(-1)^{cbx_{1,i}}$ to $|\psi\rangle$. Thus, an equivalent way of obtaining the same mixed state is to perform the following operations:

- Measure $c(1-b)$
- Apply the phase $(-1)^{cbx_{1,i}}$

This observation gives rise to our simulator. For any x query, if the corresponding y OTM has not been queried yet, the simulator will choose a random $y_{i,0}, y_{i,1}$. Then, if $y_{i,0} = 0$, it will measure $c(1-b)$ from the query, obtaining a value d.

Next, it needs to apply the phase $(-1)^{cbx_{1,i}}$. This would seem to require choosing $x_{1,i}$. However, if $d = c(1-b)$ happened to be 1, then we know that the superposition is only over $c = 1$ and $b = 0$. In particular, in the $d = 1$ case, no phase needs to be added and the final state of after the query is independent of $x_{1,i}$.

If in addition to $d = 1$, it were also the case that $y_{1,i} = 1$, then we know that $x_{1,i}$ is needed to compute z_1. This means, in the $y_{0,i} = 0, y_{1,i} = 1, d = 1$ case, the adversary's view will actually be independent of z_1. The simulator can therefore choose its own random z_1 independent of the real z_1, and simulate the remaining $x_{1,i'}, y_{1,i'}$ values accordingly. Since the adversary's view is independent of the real z_1, this will perfectly simulate the adversary's view

Hence, at this point the simulator can simulate the entire view of the adversary by simply making a classical query to obtain z_0. By an analogous argument, if $y_{0,i} = 1, y_{1,i} = 0$, we will measure $e = cb$. Provided $e = 1$, the adversary's view will be independent of z_0, and hence we can make a single classical query to obtain z_1. By symmetry, we can apply an analogous procedure for each y query that appeared before the corresponding x query.

The good news is that we are guaranteed to have many instances where $y_{0,i} = 0, y_{1,i} = 1$ or $y_{0,i} = 0, y_{1,i} = 1$, meaning there are many opportunities to obtain a $d = 1$ or $e = 1$ measurement.

The bad news is that we might get unlucky and get $d = 0$ or $e = 0$ every single time. In these cases, the post-query state for each query is in superposition over $(c, b) \in \{(1, 1), (0, 0), (0, 1)\}$ or $(c, b) \in \{(1, 0), (0, 0), (0, 1)\}$, depending on which measurement we performed. In these cases, the phase will require both $x_{0,i}$ and $x_{1,i}$. Therefore, if no measurement yielded $d = 0$ or $e = 0$, exact simulation will ultimately require knowing both z_0 and z_1.

We show, however, that conditioned on never hitting a good measurement outcome, the view of the adversary is actually statistically independent of *both* z_0, z_1. The idea is that the adversary, when making the query, does not know the values of $y_{0,i}, y_{1,i}$. In particular, it does not know whether we will measure $c(1-b)$ or cb. If the adversary places significant "weight" on $cb = 1$, there is a good chance we will measure $e = 1$. Likewise, if the adversary places significant "weight" on $c(1-b) = 1$, there is a good chance we will measure $d = 1$.

Thus, the only way to have a reasonable chance of having all such measurements result in zero is if the total "weight" on $c = 1$ terms is small. But these $c = 1$ terms are the only terms with non-trivial phase, since the phase is always 1 in the case $c = 0$. Therefore, intuitively, having small weight on $c = 1$ means you cannot perfectly learn the relevant query outputs. The intuition is that over all queries these errors compound to the point that by the end, the adversary's view is completely independent of z_0, z_1.

We show this to be the case, but the argument is quite delicate, owing to the fact that it is difficult to characterize how these errors from different queries interact with each other. The bulk of our proof goes into this piece of the analysis.

The result is a simulator that roughly does the following: for early x queries with $y_{0,i} = y_{1,i}$, the simulator simply runs the on-the-fly simulation. On the

other hand, if $y_{0,i} \neq y_{1,i}$, measure $d = c(1-b)$ or $e = cb$ according to the above. The first time such a measurement gives 1, make a classical query to the appropriate z, and make a random guess for the other z value. Simulate the rest of the queries using these z's. Similarly handle y queries. If after a large enough number of queries no measurement gives a 1, then simply choose random values for both z_0, z_1 and simulate accordingly.

1.4 Discussion

Our result can be viewed as an application of the measurement principle in quantum mechanics. Indeed, during simulation, we can replace the classical OTM inputs with quantum states. In this case, when the adversary makes a query, he is learning something about the one-time memory system and must leave behind some effect. Our simulator must then be able to interpret these effects in order to decide which input the adversary is interested in. This high-level idea has a long history, being used to prove various impossibilities for unconditionally secure quantum protocols [LC97, May97, Nay99], as well as more recent positive results on quantum indifferentiability [Zha19]. We note that the technical details underlying these various results are all quite different, owing to the very different goals of these works.

We also note that our setting, though belonging broadly to the family superposition attacks on classical cryptosystems, is fundamentally different than that of the prior work [KM10, Zha12, BZ13a, BZ13b, DFNS14, KLLN16, Zha16, GHS16, ATTU16, GYZ17, SS17], giving rise to entirely different techniques. Prior work in this space was concerned about attaining concrete security notions — unforgeable signatures, semantically secure encryption — in the setting of superposition attacks, mostly in the computational setting. In contrast, our goal is to realize general composable functionalities in and information-theoretic setting.

1.5 Concurrent and Independent Work

Concurrently and independently to our work, Ebrahimi et al. [ECKM20] examine superposition attacks against oblivious transfer (OT). OT is the multi-party computation analog of one-time memories (OTM), and has the same ideal functionality as OTM. As in our work, they observe that applying the Deutsch-Jozsa algorithm to the ideal OT functionality allows one to learn the parity of the two input bits. They then present a protocol which they show is immune to the Deutsch-Jozsa algorithm, and conjecture security against all attacks.

On the other hand, Ebrahimi et al claim that, in a superposition access model, the ideal functionality must accept quantum superpositions and a classical ideal functionality is impossible[4]. Since quantum access to the ideal functionality allows one to learn the parity of the input bits using Deutsch-Jozsa,

[4] Concretely, the authors state: "[A] real world protocol may be executed in superposition by the adversary. Therefore to have a meaningful security model, we need to consider an ideal protocol that will be run in superposition too"

their conclusion is then to reject the real/ideal paradigm in favor of a distinguishability definition. Their definition roughly requires that the parity of the sender's bits is hidden to the receiver.

Our work makes progress toward refuting their claim: our OTM immunizer can also be applied to OT, compiling any "quantum accessible OT" (where the ideal functionality allows quantum access) into a "classically accessible OT" (where the ideal functionality only allows classical access). In Appendix B, we additionally show how a common OT reorientation protocol gives rise to a quantum accessible OT, assuming certain correlations between the sender and receiver. Applying our immunizer gives a classically accessible OT from such correlations, despite the receiver's ability to query the sender on superposition. We leave as an open question achieving quantum accessible OT in the standard model.

2 Preliminaries

A binary string $\mathbf{x}$ is represented as $x_1 x_2 \cdots x_\ell$ where ℓ is the length of the string. For a matrix $\boldsymbol{\rho}$, $|\boldsymbol{\rho}|_1$ is the trace norm of $\boldsymbol{\rho}$. For any two matrices $\boldsymbol{\rho}, \boldsymbol{\sigma}$ of the same dimension, their trace distance is defined as $T(\boldsymbol{\rho}, \boldsymbol{\sigma}) = \frac{1}{2}\mathsf{Tr}\left[\sqrt{(\boldsymbol{\rho}-\boldsymbol{\sigma})^2}\right] = \frac{1}{2}|\boldsymbol{\rho}-\boldsymbol{\sigma}|_1$. $T(\boldsymbol{\rho}, \boldsymbol{\sigma})$ is the quantum generalization of total variation distance which gives the upper bound of advantage that any quantum algorithm can achieve for distinguishing $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$. We say two quantum states $\boldsymbol{\rho}_\lambda, \boldsymbol{\sigma}_\lambda$ are **statistically close**, then there exists a negligible function $\mathsf{negl}(\cdot)$, such that $T(\boldsymbol{\rho}_\lambda, \boldsymbol{\sigma}_\lambda)$ is bounded by $\mathsf{negl}(\lambda)$.

2.1 Quantum Computation

We briefly review the basics of quantum computation. A quantum system Q corresponds to a complex Hilbert space. The state of the system $|\psi\rangle$ is a complex unit vector in Q. We will normally think of $|\psi\rangle$ as a column vector, and its conjugate transpose will be denoted $\langle\psi|$. The inner product between $|\psi\rangle$ and $|\phi\rangle$ is thus $\langle\phi|\psi\rangle$.

Basic Measurements. A pure state $|\phi\rangle$ can be measured; the measurement outputs the value x with probability $|\langle x|\phi\rangle|^2$. The normalization of $|\phi\rangle$ ensures that the distribution over x is indeed a probability distribution. After measurement, the state "collapses" to the state $|x\rangle$. Notice that subsequent measurements will always output x, and the state will always stay $|x\rangle$.

If $Q = Q_0 \times Q_1$, we can perform a **partial measurement** in the system Q_0 or Q_1. If $|\phi\rangle = \sum_{x\in B_0, y\in B_1} \alpha_{x,y}|x, y\rangle$, partially measuring in Q_0 will give x with probability $p_x = \sum_{y\in B_1} |\alpha_{x,y}|^2$. $|\phi\rangle$ will then collapse to the state $\sum_{y\in B_1} \frac{\alpha_{x,y}}{\sqrt{p_x}}|x, y\rangle$. In other words, the new state has support only on pairs of the form (x, y) where x was the output of the measurement, and the weight on each pair is proportional to the original weight in $|\phi\rangle$. Notice that subsequent partial measurements over Q_0 will always output x, and will leave the state unchanged.

We can also perform more complex partial measurements. For example, consider a classical function $f : B_0 \to B_1$. We can initialize extra qubits to 0, and then perform the map

$$|x\rangle \otimes |y\rangle \mapsto |x\rangle \otimes |y + f(x)\rangle$$

If we start with the state $|\phi\rangle = \sum_{x \in B_0} \alpha_x |x, 0\rangle$, the resulting state will be $\sum_{x \in B_0} \alpha_x |x, f(x)\rangle$. Finally, we can measure the extra registers, obtaining $y = f(x)$. The state then collapses to $\propto \sum_{x:f(x)=y} \alpha_x |x, y\rangle$; at this point the two registers are un-entangled, so we can discard the y register, resulting in the state $|\phi_y\rangle \propto \sum_{x:f(x)=y} \alpha_x |x\rangle$. We will call the above procedure "measuring $f(x)$".

Quantum Queries. If a quantum algorithm makes queries to some function f, there are two scenarios we will consider. In one, the oracle only accepts classical queries, in which case the algorithm must measure its query. The other case is if the oracle accepts quantum queries. Here, the oracle accepts a quantum state, and applies the unitary U_f as defined above.

Mixed states. A quantum system may, for example, be in a pure state $|\phi\rangle$ with probability 1/2, and a different pure state $|\psi\rangle$ with probability 1/2. This can occur, for example, if a partial measurement is performed on a product system, resulting in a distribution over pure states.

This probability distribution on pure states cannot be described by a pure state alone. Instead, we say that the system is in a **mixed state**. The statistical behavior of a mixed state can be captured by **density matrix**. If the system is in pure state $|\phi_i\rangle$ with probability p_i, then the density matrix for the system is defined as $\rho = \sum_i p_i |\phi_i\rangle\langle\phi_i|$.

The density matrix is therefore a positive semi-definite complex Hermitian matrix with rows and columns indexed by the elements of B. The density matrix for a pure state $|\phi\rangle$ is given by the rank-1 matrix $|\phi\rangle\langle\phi|$. Any probability distribution over classical states can also be represented as a density matrix, namely the diagonal matrix where the diagonal entries are the probability values.

Measurements on density matrices. Suppose we were to measure some function f of x on $\rho = \sum_{x,x'} \rho_{x,x'} |x\rangle\langle x'|$, which we write somewhere outside the system ρ. Then the state left behind can be written as

$$\sum_{x,x':f(x)=f(x')} \rho_{x,x'} |x\rangle\langle x'| \quad \text{normalized}$$

That is, a measurement has the effect of zeroing out the entries of the density where the outcome of the measurement in the row and column differ.

2.2 One-Time Memory

In this section, we give a formal definition of one-time memory functionality, which follows from [BGS13] and [BGZ18]. The one-time memory (OTM) functionality involves two parties, the sender and the receiver. The sender takes two

input bits z_0, z_1, generates and sends a specifically implemented hardware to the receiver; the receiver upon receiving this hardware, can run a algorithm with this hardware and an input bit b to get the bit z_b; informally, after the receiver getting one of the z_b, the hardware "self-destructs". Therefore the receiver can only query it once and know exactly one of z_0 and z_1.

Definition 1. *A classical OTM (c-OTM) model for two binary strings $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$ is the following, for every quantum algorithm $\mathcal{A}$ having the c-OTM of $\mathbf{z}_0, \mathbf{z}_1$, it is equivalent to interact with the ideal functionality $\mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{cOTM}}$ (Functionality 1),*

Functionality 1 Ideal Functionality $\mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{cOTM}}$

1: Create **:** Upon inputs $(\mathbf{z}_0, \mathbf{z}_1)$ where both are bit strings of length n, the sender stores $(z_{0,i}, z_{1,i})$ for each $1 \leq i \leq n$; the sender also prepares a table $\{m_i\}_{i=1}^n$ where each m_i is initialized as 0 indicating if the i-th input pair has been queried or not.
2: Execute **:** Upon classical inputs b, i from the receiver, the sender first checks if $m_i = 0$. If not, the information about $z_{0,i}$ and $z_{1,i}$ has already been queried by the receiver and has been deleted by the sender. The sender simply sends back $\perp$. Otherwise, the sender sends $z_{b,i}$ to the receiver, marks m_i as 1 and deletes $z_{0,i}$ and $z_{1,i}$.

The interaction between a quantum algorithm $\mathcal{A}$ and $\mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{cOTM}}$ is defined as the follows: at the beginning, the sender executes Create with inputs $\mathbf{z}_0, \mathbf{z}_1$; then as the receiver, $\mathcal{A}$ can make polynomial number of Execute queries with classical inputs b, i and do quantum computation. In other words, given the c-OTM of $\mathbf{z}_0, \mathbf{z}_1$, for every $1 \leq i \leq n$, every algorithm can learn either $z_{0,i}$ or $z_{1,i}$. We denote the final density matrix of $\mathcal{A}$ as $(\mathcal{A} \Longleftrightarrow \mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{cOTM}})$ which takes all possible randomness of $\mathcal{A}$ and $\mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{cOTM}}$.

Definition 2. *A quantum-accessible OTM (q-OTM) model for two binary strings $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$ is the following, for every quantum algorithm $\mathcal{A}$ having the q-OTM of $\mathbf{z}_0, \mathbf{z}_1$, it is equivalent to interact with the ideal functionality $\mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{qOTM}}$ (Functionality 2),*

Similar to the definition of c-OTM, we denote the final density matrix of $\mathcal{A}$ in the interaction as $(\mathcal{A} \Longleftrightarrow \mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{qOTM}})$ which takes all possible randomness of $\mathcal{A}$ and $\mathcal{F}_{\mathbf{z}_0,\mathbf{z}_1}^{\mathsf{qOTM}}$.

3 Immunizing Quantum One-Time Memories

In this section, our goal is to construct a c-OTM where the ideal functionality is classical, even though the adversary is allowed to be quantum and make quantum superposition queries. We will assume we are working in the q-OTM model, in other words, the algorithm has access to q-OTM instances which are modeled

Functionality 2 Ideal Functionality $\mathcal{F}^{\mathsf{qOTM}}_{\mathbf{z}_0,\mathbf{z}_1}$

1: Create : Upon inputs $(\mathbf{z}_0, \mathbf{z}_1)$ where both are bit strings of length n, the sender stores $(z_{0,i}, z_{1,i})$ for each $1 \leq i \leq n$; the sender also prepares a table $\{m_i\}_{i=1}^n$ where each m_i is initialized as 0 indicating if the i-th input pair has been queried or not.
2: Execute : Upon a quantum state $\sum_{b,c} \alpha_{b,c}|b,c\rangle$ and a classical i from the receiver, the sender first checks if $m_i = 0$. If not, the information about $z_{0,i}$ and $z_{1,i}$ has already been queried by the receiver and has been deleted by the sender. The sender simply does nothing. Otherwise, the sender applies the unitary $U_{z_{0,i},z_{1,i}} : |b,c\rangle \to (-1)^{c \cdot z_{b,i}}|b,c\rangle$. It then sends the quantum state back to the receiver, marks m_i as 1 and deletes both $z_{0,i}$ and $z_{1,i}$.

as the ideal functionality $\mathcal{F}^{\mathsf{qOTM}}$. We will show how to immunize such an OTM to result in a classical ideal functionality.

In this section, we will first give the construction for single bit z_0, z_1 and prove the security. Then we will extend the construction to the general case, for two arbitrary strings $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$ in subsection 3.2.

3.1 c-OTM construction for single input

The Scheme. For $z_0, z_1 \in \{0,1\}$, the sender chooses 4 uniformly random strings $\mathbf{x}_0, \mathbf{y}_0, \mathbf{x}_1, \mathbf{y}_1 \in \{0,1\}^\lambda$ such that for all b, $z_b = \langle \mathbf{x}_b, \mathbf{y}_b \rangle$. The sender prepares 2λ q-OTM instances of the following input pairs $(x_{0,j}, x_{1,j})$ and $(y_{0,j}, y_{1,j})$ for $1 \leq j \leq \lambda$. The functionality of our construction is equivalent to the following (Functionality 3).

Theorem 1. *The above construction for bit inputs is a c-OTM.*

First, let us look at the ideal functionality (Functionality 3) of our construction for bit inputs.

Given an algorithm $\mathcal{A}$, $\mathcal{A}$'s interaction with Functionality 3 and its computation can be modeled as a sequence of Execute (of inputs a quantum state, j and tag), and unitary transformation.

Lemma 1. *There exists an efficient simulator* Sim_0*, for every fixed* z_0, z_1*, for every quantum algorithm* $\mathcal{A}$*,*

$$T\left(\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\mathsf{Sim}_0(\mathcal{A}) \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right)\right) = 0$$

Moreover, $\mathsf{Sim}_0(\mathcal{A})$ *has the following properties: (1) every* (j, tag) *pair will be queried at most once; (2) every* $1 \leq j \leq \lambda$*,* (j,x) *is always queried before* (j,y)*. In other words, if 1) both are queried, then* (j,x) *is queried before* (j,y)*; 2) only one is queried, then it is* (j,x)*; otherwise, neither is queried.*

Remark. In the rest of the paper, all simulators only monitors/manipulates communications between an adversary and an ideal functionality. The notation of composing two simulators, $(\mathsf{Sim}_1 \circ \mathsf{Sim}_0)(\mathcal{A}) = \mathsf{Sim}_1(\mathsf{Sim}_0(\mathcal{A}))$, denotes Sim_1

Functionality 3 The construction for bits $\mathcal{F}^*_{z_0,z_1}$

1: Create **:** Upon input bits (z_0, z_1),
   1. The sender chooses 4 random strings $\mathbf{x}_0, \mathbf{y}_0, \mathbf{x}_1, \mathbf{y}_1 \in \{0,1\}^\lambda$ such that for $z_0 = \langle \mathbf{x}_0, \mathbf{y}_0 \rangle$ and $z_1 = \langle \mathbf{x}_1, \mathbf{y}_1 \rangle$.
   2. The sender stores pairs $(x_{0,j}, x_{1,j})$ and $(y_{0,j}, y_{1,j})$ for $1 \leq j \leq \lambda$. It also prepares a table $\{m_{x,j} = 0\}$ and $\{m_{y,j} = 0\}$ indicating if the j-th pair corresponding to either $\mathbf{x}$ or $\mathbf{y}$ has been queried or not.

2: Execute **:** Upon a quantum state $\sum_{b,c} \alpha_{b,c}|b,c\rangle$ and classical inputs j, tag from the receiver ($\mathsf{tag} \in \{x, y\}$), let us assume $\mathsf{tag} = x$. The same holds for $\mathsf{tag} = y$.
   1. The sender first checks if $m_{x,j} = 0$. If not, the information about $x_{0,j}$ and $x_{1,j}$ has already been queried by the receiver and has been deleted by the sender. The sender simply does nothing and sends the state back.
   2. Otherwise, the sender applies the unitary $U_j : |b,c\rangle \to (-1)^{c \cdot x_{b,j}}|b,c\rangle$. It then sends the quantum state back to the receiver, marks $m_{x,j}$ as 1 and deletes both $x_{0,j}$ and $x_{1,j}$.

monitors/manipulates communications between $\mathsf{Sim}_0(\mathcal{A})$ and the ideal functionality.

We are proving these two properties separately. By composing the following two simulators Sim' and Sim'', we complete the proof.

Claim. Every (j, tag) pair will only be queried at most once.

Proof. If there are two Execute with inputs $(|\phi_1\rangle, j, \mathsf{tag})$ and $(|\phi_2\rangle, j, \mathsf{tag})$, we can simply remove the second Execute and the computation will give exactly the same result. By the definition, for the second $\mathsf{Execute}(|\phi_2\rangle, j, \mathsf{tag})$, it does nothing to $|\phi_2\rangle$.

In other words, there exists an efficient simulator Sim'. It runs $\mathcal{A}$ as a subroutine and records all queries from $\mathcal{A}$: for a query $(|\phi\rangle, j, \mathsf{tag})$, if (j, tag) has been recorded, it does nothing and returns the state; otherwise, it records the pair (j, tag) in its database, forwards the query to $\mathcal{F}^*_{z_0,z_1}$.Execute and returns it back to $\mathcal{A}$. Therefore, for every $\mathcal{A}$, $(\mathsf{Sim}'(\mathcal{A}) \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}) = (\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1})$. Every (j, tag) is queried at most once by $\mathsf{Sim}'(\mathcal{A})$. □

Claim. For every $1 \leq j \leq \lambda$, (j, x) is always queried before (j, y). In other words, if 1) both are queried, then (j, x) is queried before (j, y); 2) only one is queried, then it is (j, x); otherwise, neither is queried.

Proof. For a fixed j, we know that $x_{0,j}, x_{1,j}$ and $y_{0,j}, y_{1,j}$ are symmetric because they have the same distribution and they contribute equally to the inner product $\langle \mathbf{x}_0, \mathbf{y}_0 \rangle$ and $\langle \mathbf{x}_1, \mathbf{y}_1 \rangle$. Therefore, we can always force the first query to be the form of $(|\phi\rangle, j, x)$.

There exists an efficient simulator Sim''. It runs $\mathcal{A}$ as a subroutine and records if the j-th query needs to be flipped, denoted as a variable w_j initialized as 0. If $w_j = 0$, the j-th query has not been made yet; otherwise, $w_j = 1, 2$ denotes (j, x) or (j, y) is queried first respectively.

For a query $(|\phi\rangle, j, \mathsf{tag})$, if w_j is 0, it lets $w_j = 1$ if $\mathsf{tag} = x$ and lets $w_j = 2$ if $\mathsf{tag} = y$. It then forwards the query, and if $w_j = 2$, it flips tag before forwarding it. Therefore, for every $1 \leq j \leq \lambda$, $\mathsf{Sim}''(\mathcal{A})$ always queries (j, x) before (j, y). We also have, $(\mathsf{Sim}''(\mathcal{A}) \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}) = (\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1})$. □

Next we will show that even if we switch to another ideal functionality which gives both $\mathbf{y}_0, \mathbf{y}_1$ to the adversary, our construction is still a c-OTM.

Lemma 2. *There exists an efficient simulator* Sim_1*, for every fixed* z_0, z_1*, for every quantum algorithm* $\mathcal{A}$*, let* $\mathcal{B} = \mathsf{Sim}_0(\mathcal{A})$*,*

$$T\left(\left(\mathcal{B} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\mathsf{Sim}_1(\mathcal{B}) \Longleftrightarrow \mathcal{F}^4_{z_0,z_1}\right)\right) = 0$$

Moreover, $\mathsf{Sim}_1(\mathcal{B})$ *queries* $\mathcal{F}^4_{z_0,z_1}.\mathsf{Execute}$ *on* j *in the order from* 1 *to* λ*: that is, it queries on each input* j *exactly once and if* $j < k$*, then* j *is queried before* k*.* $\mathcal{F}^4_{z_0,z_1}$ *is defined as follows (Functionality 4).*

In Functionality 4, as we know $\mathcal{B}$ will always query (j, x) before (j, y), $\mathsf{Execute}$ always answers the quantum query for $x_{0,j}$ and $x_{1,j}$ as in Functionality 3 and together sends both $y_{0,j}$ and $y_{1,j}$ back to the receiver.

Functionality 4 The construction for bits $\mathcal{F}^4_{z_0,z_1}$

1: Create **:** Upon input bits (z_0, z_1),
   1. The sender chooses 4 random strings $\mathbf{x}_0, \mathbf{y}_0, \mathbf{x}_1, \mathbf{y}_1 \in \{0,1\}^\lambda$ such that for $z_0 = \langle \mathbf{x}_0, \mathbf{y}_0 \rangle$ and $z_1 = \langle \mathbf{x}_1, \mathbf{y}_1 \rangle$.
   2. The sender stores pairs $(x_{0,j}, x_{1,j})$ and $(y_{0,j}, y_{1,j})$ for $1 \leq j \leq \lambda$. It also prepares a table $\{m_j = 0\}$ indicating if the j-th pair corresponding to $\mathbf{x}$ and $\mathbf{y}$ has been queried or not.

2: $\mathsf{Execute}$ **:** Upon a quantum state $\sum_{b,c} \alpha_{b,c}|b, c\rangle$ and classical input j.
   1. The sender first checks if $m_j = 0$. If not, the information has already been queried by the receiver and has been deleted by the sender. The sender simply does nothing and sends the state back.
   2. Otherwise, the sender applies the unitary $U_j : |b, c\rangle \to (-1)^{c \cdot x_{b,j}}|b, c\rangle$. It then sends the quantum state and two *classical bits* $y_{0,j}, y_{1,j}$ back to the receiver, marks m_j as 1 and deletes $x_{0,j}, x_{1,j}, y_{0,j}, y_{1,j}$.

Combining with Lemma 1, we have the following corollary,

Corollary 1. *There exists an efficient simulator* $\widetilde{\mathsf{Sim}}_1$*, for every fixed* z_0, z_1*, for every quantum algorithm* $\mathcal{A}$*,*

$$T\left(\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\widetilde{\mathsf{Sim}}_1(\mathcal{A}) \Longleftrightarrow \mathcal{F}^4_{z_0,z_1}\right)\right) = 0$$

Moreover, $\widetilde{\mathsf{Sim}}_1(\mathcal{A})$ *queries* $\mathcal{F}^4_{z_0,z_1}.\mathsf{Execute}$ *on* j *in the order from* 1 *to* λ*.*

Proof (Corollary 1). Let $\widetilde{\mathsf{Sim}}_1 = \mathsf{Sim}_1 \circ \mathsf{Sim}_0$. We have, for every $\mathcal{A}$,

$$\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right) = \left(\mathsf{Sim}_0(\mathcal{A}) \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right) = \left(\widetilde{\mathsf{Sim}}_1(\mathcal{A}) \Longleftrightarrow \mathcal{F}^4_{z_0,z_1}\right)$$

□

An algorithm interacting with $\mathcal{F}^4$ of input z_0, z_1 can be modeled as a sequence of unitary operations V_i and oracle access to $U_j : |b,c\rangle \to (-1)^{c\cdot x_{b,j}}|b,c\rangle$ together with auxiliary classical outputs $y_{0,j}, y_{1,j}$ where $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ are chosen uniformly at random such that $\langle \mathbf{x}_0, \mathbf{y}_0\rangle = z_0, \langle \mathbf{x}_1, \mathbf{y}_1\rangle = z_1$.

Definition 3. *Define the oracle O_j as a unitary U_j and two classical outputs $y_{0,j}, y_{1,j}$,*

$$U_j|b,c,\mathsf{aux}\rangle \to (-1)^{c\cdot x_{b,j}}|b,c,\mathsf{aux}\rangle$$

Proof (Lemma 2). We know that $\mathcal{B} = \mathsf{Sim}_0(\mathcal{A})$ has the following properties: (1) every (j,tag) pair will be queried at most once; (2) every $1 \le j \le \lambda$, (j,x) is always queried before (j,y).

Our construction of Sim_1 contains two parts: Sim'_1 and Sim''_1. Define simulator $\mathsf{Sim}'_1(\mathcal{B})$ as:

1. Sim'_1 runs $\mathcal{B}$ as a subroutine, every time $\mathcal{B}$ wants to access $\mathcal{F}^*_{z_0,z_1}.\mathsf{Execute}$ with input $(|\phi\rangle, j, \mathsf{tag})$,
   - If $\mathsf{tag} = x$, Sim'_1 sends the input to $\mathcal{F}^4_{z_0,z_1}.\mathsf{Execute}$ and gets $|\phi\rangle$ updated (according to $x_{0,j}, x_{1,j}$) as well as classical $y_{0,j}, y_{1,j}$ back. Sim'_1 stores $y_{0,j}, y_{1,j}$ and sends the updated quantum state back to $\mathcal{B}$.
   - Otherwise $\mathsf{tag} = y$, Sim'_1 has $y_{0,j}, y_{1,j}$ stored so it can simulate what $\mathcal{F}^*.\mathsf{Execute}$ does. Sim'_1 generates and applies the unitary $U_j : |b,c\rangle \to (-1)^{c\cdot y_{b,j}}|b,c\rangle$ to $|\phi\rangle$, sends $U_j|\phi\rangle$ back to $\mathcal{B}$ and discards $y_{0,j}, y_{1,j}$.
2. Sim'_1 keeps running $\mathcal{B}$ and outputs what $\mathcal{B}$ outputs.

The only difference between Functionality 3 and 4 is about queries on $\mathsf{tag} = y$. For every access of the form $\mathsf{Execute}$ with input $(|\phi\rangle, j, \mathsf{tag} = y)$, since Sim'_1 already stored $y_{0,j}$ and $y_{1,j}$ after it queries (j,x), it knows what the unitary is and perfectly simulates what Functionality 3 does. Therefore Sim'_1 perfectly simulates the output of $\mathcal{B}$, i.e., $(\mathcal{B} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}) = (\mathsf{Sim}'_1(\mathcal{B}) \Longleftrightarrow \mathcal{F}^4_{z_0,z_1})$.

Next we show there exists an efficient simulator $\mathsf{Sim}''_1 = \mathsf{Sim}'' \circ \mathsf{Sim}'$ such that $\mathsf{Sim}''_1(\mathsf{Sim}'_1(\mathcal{B}))$ queries O_j exactly once for every j and O_j is queried before O_k if $j < k$.

Claim. Each $O_j (1 \le j \le \lambda)$ is queried exactly once.

Proof. As we know each O_j is queried at most once, we only need to show they are queried at least once. If O_j is not queried by an algorithm, it can simply make a classical query to O_j and discard everything it gets. It will have the same output.

The simulator Sim' simply records all queries j made by $\mathcal{B}$ so far. When $\mathcal{B}$ halts, it queries all j that has not been queried before with any classical input and discard them. □

Claim. O_j is queried before O_k if $j < k$.

Proof. Since each entry in $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ is symmetric and contributes equally to the inner product, we can simply rename it. In other words, the simulator Sim'' always maps the first query j_1 to 1, the second query j_2 to 2 and so on. The algorithm queries O'_{j_1} in the first round, which is actually O_1, it queries $O'_{j_2} = O_2$ in the second round and so on. The distributions of the unitary it gets access are identical. Therefore its output remains the same density matrix. □

Let $\mathsf{Sim}''_1 = \mathsf{Sim}'' \circ \mathsf{Sim}'$ and $\mathsf{Sim}_1 = \mathsf{Sim}''_1 \circ \mathsf{Sim}'_1$. We conclude Lemma 2. □

With the above lemma and corollary, we model the computation of an algorithm $\mathcal{A}$ interacting with Functionality 4 as:

1. It starts with an all-zero quantum state $\boldsymbol{\rho}_0$ of polynomial size.
2. At step $i \in [\lambda]$, the internal quantum state is $\boldsymbol{\rho}_{i-1}$. It applies a unitary $V_i \boldsymbol{\rho}_{i-1} V_i^\dagger$, and gets access to O_i. The state $\boldsymbol{\rho}_i$ becomes

$$\boldsymbol{\rho}_i = U_i V_i \boldsymbol{\rho}_{i-1} V_i^\dagger U_i^\dagger \otimes |y_{0,i}, y_{1,i}\rangle\langle y_{0,i}, y_{1,i}|$$

3. Finally it applies a measurement over $\boldsymbol{\rho}_\lambda$.

The overall density matrix also takes randomness over the uniform choice of $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ such that $\langle \mathbf{x}_0, \mathbf{y}_0 \rangle = z_0, \langle \mathbf{x}_1, \mathbf{y}_1 \rangle = z_1$. In other words, $\mathbf{x}_b, \mathbf{y}_b$ are repeatedly sampled uniformly at random from $\{0,1\}^\lambda$, until $\langle \mathbf{x}_b, \mathbf{y}_b \rangle = z_b$. Both U_i and $y_{0,i}, y_{1,i}$ depend on z_0, z_1. There is a joint distribution over these four vectors and z_0, z_1. Next we show that the way we generate $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ can be slightly altered. We generate the first $\lambda/2$ bits of these vectors unconditionally and then repeatedly sample the rest of the vectors until the inner product are z_0 and z_1. The statistical distance remains negligible (see Lemma 3).

Lemma 3. *Assume λ is even. Fix a $z \in \{0,1\}$, consider the following ways of generating vectors:*

1. *$\mathbf{x}, \mathbf{y}$ are repeatedly sampled uniformly at random from $\{0,1\}^\lambda$, until $\langle \mathbf{x}, \mathbf{y} \rangle = z$.*
2. *$\mathbf{x}'$ and $\mathbf{y}'$ are sampled as follows: sample and fix $\mathbf{x}'_<, \mathbf{y}'_<$ uniformly at random from $\{0,1\}^{\lambda/2}$, then $\mathbf{x}'_>, \mathbf{y}'_>$ are repeatedly sampled uniformly at random from $\{0,1\}^{\lambda/2}$, until $\langle \mathbf{x}'_<, \mathbf{y}'_< \rangle + \langle \mathbf{x}'_>, \mathbf{y}'_> \rangle = z$. Let $\mathbf{x}' = \mathbf{x}'_< || \mathbf{x}'_>$ and $\mathbf{y}' = \mathbf{y}'_< || \mathbf{y}'_>$.*

The statistical distance between these vectors are negligible in λ.

$$\Delta\left((\mathbf{X}, \mathbf{Y}), (\mathbf{X}', \mathbf{Y}')\right) \leq 2^{-\Omega(\lambda)}$$

Here $\mathbf{X}, \mathbf{Y}$ are the corresponding random variables for $\mathbf{x}, \mathbf{y}$. For the case λ is odd, the same conclusion holds by letting $|\mathbf{x}'_<| = |\mathbf{y}'_<| = \lfloor \lambda/2 \rfloor$ and $|\mathbf{x}'_>| = |\mathbf{y}'_>| = \lceil \lambda/2 \rceil$.

Proof. The proof is in Appendix A.1. A direct calculation gives the lemma. □

By Lemma 3, $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ are sampled according to the second method: the first half of the vectors are sampled completely at random, and the second half are generated uniformly to satisfy the inner product constraint. More formally, $\mathbf{x}_0$ and $\mathbf{y}_0$ are sampled as follows: sample $\mathbf{x}_<, \mathbf{y}_<$ uniformly at random from $\{0,1\}^{\lambda/2}$, then $\mathbf{x}_>, \mathbf{y}_>$ are repeatedly sampled uniformly at random from $\{0,1\}^{\lambda/2}$, until $\langle \mathbf{x}_<, \mathbf{y}_< \rangle + \langle \mathbf{x}_>, \mathbf{y}_> \rangle = z_0$. Let $\mathbf{x}_0 = \mathbf{x}_< || \mathbf{x}_>$ and $\mathbf{y}_0 = \mathbf{y}_< || \mathbf{y}_>$. The same for $\mathbf{x}_1, \mathbf{y}_1$. Therefore every algorithm making queries $\mathsf{Execute}$ with input j in the order from 1 to λ can not distinguish if it is interacting with Functionality 4 or Functionality 5.

Therefore, combining Lemma 2 and 3, we have the following lemma.

Lemma 4. *There exists an efficient simulator* Sim_1 *(the same simulator in Lemma 2), for every fixed* z_0, z_1*, for every quantum algorithm* $\mathcal{A}$*, let* $\mathcal{B} = \mathsf{Sim}_0(\mathcal{A})$*.* $\mathsf{Sim}_1(\mathcal{B})$ *queries* $\mathcal{F}^5_{z_0,z_1}.\mathsf{Execute}$ *on* j *in the order from* 1 *to* λ*, and*

$$T\left(\left(\mathcal{B} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\mathsf{Sim}_1(\mathcal{B}) \Longleftrightarrow \mathcal{F}^5_{z_0,z_1}\right)\right) \leq 2^{-\Omega(\lambda)}$$

Functionality 5 $\mathcal{F}^5_{z_0,z_1}$

1: Create **:** It is the same as Create in Functionality 4 but $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ are sampled using the second method above on the fly (in $\mathsf{Execute}$ procedure). It stores z_0, z_1 and two partial inner product $s_0 = 0, s_1 = 0$.

2: $\mathsf{Execute}$ **:** Upon input a quantum state and a classical j,

   1. If $1 \leq j \leq \lambda/2$, it samples $x_{0,j}, x_{1,j}, y_{0,j}, y_{1,j}$ uniformly at random, and updates $s_0 \leftarrow s_0 \oplus x_{0,j}y_{0,j}$, $s_1 \leftarrow s_1 \oplus x_{1,j}y_{1,j}$.
   2. Else if $j = \lambda/2 + 1$, it samples and stores the second half of the vectors $\mathbf{x}_{0,>}, \mathbf{x}_{1,>}, \mathbf{y}_{0,>}, \mathbf{y}_{1,>}$ uniformly at random such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>} \rangle + s_0 = z_0$ and $\langle \mathbf{x}_{1,>}, \mathbf{y}_{1,>} \rangle + s_1 = z_1$.

   The rest are the same as $\mathsf{Execute}$ in Functionality 4, which updates the input quantum state, gives back $y_{0,j}, y_{1,j}$ and deletes $x_{0,j}, x_{1,j}, y_{0,j}, y_{1,j}$.

To show this construction is a c-OTM, fixing z_0, z_1, we need to show that the final density matrix $\boldsymbol{\rho}_\lambda$ (taken randomness over $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$) can be simulated by only one single classical access to the function $b \to z_b$ which is exactly the ideal functionality of c-OTM. By proving the lemma below, we finish our proof for Theorem 1.

Lemma 5. *There exists an efficient simulator* Sim_2*, for every fixed* z_0, z_1*, every quantum algorithm* $\mathcal{A}$ *interacting with Functionality 5 and querying* $\mathcal{F}^5_{z_0,z_1}.\mathsf{Execute}$ *on* j *in the order from* 1 *to* λ*,*

$$T\left(\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^5_{z_0,z_1}\right), \left(\mathsf{Sim}_2(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{z_0,z_1}\right)\right) < 2^{-\Omega(\lambda)}$$

Proof. As discussed above, the computation of $\mathcal{A}$ can be modeled as a sequence of unitary V_i and oracle access to O_i. For the first $\lambda/2$ O_i, each O_i has $U_i : |b, c\rangle \to (-1)^{c \cdot x_{b,i}} |b, c\rangle$ and classical information $y_{0,i}, y_{1,i}$ where $x_{0,i}, x_{1,i}, y_{0,i}, y_{1,i}$ are completely uniformly at random. In the first $\lambda/2$ rounds,

1. The algorithm starts with an all-zero quantum state $\boldsymbol{\rho}_0$ of polynomial size, and the state for the partial inner product s_0, s_1 is $|00\rangle\langle 00|$, the overall state of the whole system (including the algorithm and the ideal functionality) is $\boldsymbol{\gamma}_0 = \boldsymbol{\rho}_0 \otimes |00\rangle\langle 00|$.
2. At the start of round $1 \le i \le \lambda/2$, the overall state is $\boldsymbol{\gamma}_{i-1}$, and the internal quantum state of the algorithm is a partial trace of $\boldsymbol{\gamma}_{i-1}$, which is $\boldsymbol{\rho}_{i-1}$.
   (a) It applies a unitary $V_i \boldsymbol{\gamma}_{i-1} V_i^\dagger$, and the resulting state is $\boldsymbol{\sigma}_i$,

$$\boldsymbol{\sigma}_i = \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_1}} \sigma^i_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_0}} |b,c,\mathsf{aux}\rangle\langle b',c',\mathsf{aux}'| \otimes |s_0,s_1\rangle\langle s_0,s_1|$$

   where b, c are the query registers, aux is $\mathcal{A}$'s private register and s_0, s_1 are the current partial inner products stored by the ideal functionality.
   (b) It then gets access to O_i. It applies $U_{x_{0,i},x_{1,i}}$ to the query registers and together gets back two classical bits $y_{0,i}, y_{1,i}$. The partial inner products s_0, s_1 are updated by $x_{b,i} y_{b,i}$ respectively.

In the above description, step (a) is a fixed unitary transformation only applied to $\mathcal{A}$'s register, which is not interesting to our case. Let us focus on step (b). To make our analysis easier, for now let us ignore the subscript or superscript 'i'. The state in (b) is,

$$\frac{1}{4} \sum_{x_0,x_1} \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_1}} (-1)^{c\cdot x_b + c'\cdot x_{b'}} \sigma_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_0}} |b,c,\mathsf{aux}\rangle\langle b',c',\mathsf{aux}'| \otimes |y_0,y_1\rangle\langle y_0,y_1| \otimes |s_0 + x_0 y_0, s_1 + x_1 y_1\rangle\langle s_0 + x_0 y_0, s_1 + x_1 y_1|$$

The phase $(-1)^{c\cdot x_b + c'\cdot x_{b'}}$ comes from unitary U_{x_0,x_1}.

With probability $1/4$, we have the classical bits $y_0 = 0, y_1 = 1$. In this case, the overall state is the following, as well as the algorithm knows the classical information $y_0 = 0, y_1 = 1$ (here we ignore $|y_0, y_1\rangle\langle y_0, y_1|$ for a better presentation).

$$\boldsymbol{\sigma} = \frac{1}{4} \sum_{x_0,x_1} \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_1}} (-1)^{c\cdot x_b + c'\cdot x_{b'}} \sigma_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_0}} |b,c,\mathsf{aux}\rangle\langle b',c',\mathsf{aux}'| \otimes |s_0, s_1 + x_1\rangle\langle s_0, s_1 + x_1|$$

Note that the only place it has x_0 is in cx_b and $c'x_{b'}$ but x_1 appears in both exponent and s_1's register. Since $c\cdot x_b + c'\cdot x_{b'} = c\cdot((1-b)\,x_0 + b\,x_1) + c'\cdot((1-b')\,x_0 + b'\,x_1)$ and x_0 is a free variable (which is independent of all registers), the summation $\sum_{x_0}(-1)^{x_0((1-b)c+(1-b')c'} = 2\cdot[(1-b)c = (1-b')c']$. We can simplify

the state as,

$$\boldsymbol{\sigma} = \frac{1}{2}\sum_{x_1} \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_1 \\ (1-b)c=(1-b')c'}} (-1)^{cbx_1+c'b'x_1} \sigma_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}' \\ s_0,s_0}} |b,c,\mathsf{aux}\rangle\langle b',c',\mathsf{aux}'| \otimes |s_0, s_1+x_1\rangle\langle s_0, s_1+x_1|$$

From preliminaries, zeroing out all entries with $(1-b)c \neq (1-b')c'$ in the density matrix is equivalent to measure $(1-b)c$. Therefore, we have the following lemma:

Lemma 6. *Applying U_{x_0,x_1} to the quantum state $\boldsymbol{\sigma}$ with $y_0 = 0$ is equivalent to measuring $(1-b)c$ and then applying the unitary $|b,c\rangle \to (-1)^{bcx_1}|b,c\rangle$.*

Similarly, applying U_{x_0,x_1} to the quantum state $\boldsymbol{\sigma}$ with $y_1 = 0$ is equivalent to measuring bc and then applying the unitary $|b,c\rangle \to (-1)^{(1-b)cx_0}|b,c\rangle$.

If $y_0 = 0, y_1 = 1$ and the measurement $(1-b)c$ gives us 1, we know that $b = 0, c = 1$ and $b' = 0, c' = 1$. In this case, $(-1)^{cbx_1+c'b'x_1}$ can be simplified as 1 and the state of the algorithm $\mathcal{A}$ is independent of x_1. However, the partial inner product s_1 is updated by adding x_1. Therefore in $\mathcal{A}$'s view, the partial inner product s_1 is completely random. In other words, the overall state of the system can be written as,

$$(p_0\boldsymbol{\rho}_0 \otimes |0\rangle\langle 0| + p_1\boldsymbol{\rho}_1 \otimes |1\rangle\langle 1|) \otimes \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|)$$

where $\boldsymbol{\rho}_b$ is the state of $\mathcal{A}$ corresponding to $s_0 = b$, the second register is s_0, the thrid register is s_1.

Similarly, if $y_0 = 1, y_1 = 0$ and the measurement bc gives 1, we know $b = c = 1$. In this case, $(-1)^{(1-b)cx_0+(1-b')c'x_0}$ can be simplified as 1. The state of the algorithm $\mathcal{A}$ is independent of x_0. However, the partial inner product s_0 is updated by adding x_0. Therefore in $\mathcal{A}$'s view, the partial inner product s_0 is completely random.

If either event above happens in the first $\lambda/2$ rounds, $\mathcal{A}$ knows at most one of s_0, s_1 and the other is completely random from its view. With Lemma 6 and the observation above, we give the construction of Sim_2 (Algorithm 6) which keeps track of whether such event happens.

Intuitively speaking, $\mathsf{Sim}_2(\mathcal{A})$ perfectly simulates $\mathcal{F}^5_{z_0,z_1}$ in the first $\lambda/2$ rounds. The binary variables G_0, G_1 denotes whether s_0 or s_1 is completely random from $\mathcal{A}$'s view. If G_0 is True, Sim_2 simply guesses the value of z_0; if G_1 is True, Sim_2 guesses z_1. Otherwise, we claim both s_0 and s_1 are random from $\mathcal{A}$'s view and therefore Sim_2 randomly samples both $\hat{z}_0$ an $\hat{z}_1$. It then keeps simulates $\mathcal{F}^5_{z_0,z_1}$.

Fixing an algorithm $\mathcal{A}$ and z_0, z_1, we define the event Good_0 as $G_0 = \mathsf{True}$ and Good_1 as $G_1 = \mathsf{True}$ when $\mathcal{F}^5_{z_0,z_1}$ or Sim_2 answers the first $\lambda/2$ queries. Finally, we define $\mathsf{Good} = \mathsf{Good}_0 \vee \mathsf{Good}_1$.

Next we are going to show the following two lemmas:

Algorithm 6 $\mathsf{Sim}_2(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{z_0,z_1}$

Require: $\mathcal{A}$ is an algorithm interacting with $\mathcal{F}^5_{z_0,z_1}$ and it queries $\mathcal{F}^5_{z_0,z_1}.\mathsf{Execute}$ on inputs j in the order from 1 to λ.

1: Sim_2 runs $\mathcal{A}$ as a subroutine and maintains partial inner products s_0, s_1. It also stores two binary variables G_0, G_1 initialized as False.
2: For the first $i \leq \lambda/2$ queries, assume the i-th query made by $\mathcal{A}$ is $(|\phi_i\rangle, i)$ and $|\phi_i\rangle$ has b, c registers for storing inputs and outputs.
   - Sim_2 samples $x_{0,i}, x_{1,i}, y_{0,i}, y_{1,i}$ uniformly at random from $\{0,1\}$ and updates s_0, s_1.
   - If $y_{0,i} = 0$ and $y_{1,i} = 1$, it first measures $(1-b)c$ and then applies the unitary $|b,c\rangle \to (-1)^{bcx_0}|b,c\rangle$. It returns the state together with $y_{0,i}, y_{1,i}$. If the measurement gives 1, set $G_1 = \mathsf{True}$.
   - If $y_{0,i} = 1$ and $y_{1,i} = 0$, it first measures bc and then applies the unitary $|b,c\rangle \to (-1)^{(1-b)cx_1}|b,c\rangle$. It returns the state together with $y_{0,i}, y_{1,i}$. If the measurement gives 1, set $G_0 = \mathsf{True}$.
   - Otherwise $y_{0,i}, y_{1,i} = 0,0$ or $1,1$, it applies $U_{x_{0,i},x_{1,i}}$ to the query and gives it back together with $y_{0,i}, y_{1,i}$.
3: After the first $\lambda/2$ queries, if $G_0 = \mathsf{True}$, Sim_2 queries $\hat{z}_1 = z_1$ and randomly samples $\hat{z}_0$. If $G_1 = \mathsf{True}$, Sim_2 queries $\hat{z}_0 = z_0$ and randomly samples $\hat{z}_1$. Otherwise, it samples both $\hat{z}_0, \hat{z}_1$ uniformly at random.
4: It samples the second half of the vectors $\mathbf{x}_{0,>}, \mathbf{x}_{1,>}, \mathbf{y}_{0,>}, \mathbf{y}_{1,>}$ uniformly at random such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>} \rangle + s_0 = \hat{z}_0$ and $\langle \mathbf{x}_{1,>}, \mathbf{y}_{1,>} \rangle + s_1 = \hat{z}_1$.
5: It answers the remaining questions for $i \geq \lambda/2 + 1$ and outputs what $\mathcal{A}$ outputs.

Lemma 7. *For every fixed z_0, z_1, every quantum algorithm $\mathcal{A}$ interacting with Functionality 5 and querying $\mathcal{F}^5_{z_0,z_1}.\mathsf{Execute}$ on j in the order from 1 to λ, let $\boldsymbol{\rho}$ denote the mixed state of $\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^5_{z_0,z_1}\right)$ conditioned on Good. Similarly, let $\boldsymbol{\rho}'$ denote the mixed state of $\left(\mathsf{Sim}_2(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{z_0,z_1}\right)$ conditioned on Good. We have, $T(\boldsymbol{\rho}, \boldsymbol{\rho}') = 0$.*

Lemma 8. *For every fixed z_0, z_1, every quantum algorithm $\mathcal{A}$ interacting with Functionality 5 and querying $\mathcal{F}^5_{z_0,z_1}.\mathsf{Execute}$ on j in the order from 1 to λ, let $\boldsymbol{\rho}$ denote the mixed state of $\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^5_{z_0,z_1}\right)$ conditioned on $\neg\mathsf{Good}$. Similarly, let $\boldsymbol{\rho}'$ denote the mixed state of $\left(\mathsf{Sim}_2(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{z_0,z_1}\right)$ conditioned on $\neg\mathsf{Good}$. There exists a constant c, if $\Pr[\neg\mathsf{Good}] > 2^{-c\lambda}$, then $T(\boldsymbol{\rho}, \boldsymbol{\rho}') < 2^{-\Omega(\lambda)}$.*

Lemma 8 says that either Good happens with overwhelming probability, or when Good does not happen, Sim_2 can almost perfectly simulates $\mathcal{F}^5_{z_0,z_1}$ by guessing both z_0 and z_1. Combining the above two lemmas, we finish the proof for Lemma 5. Because when Good happens, it can be perfectly simulated by Lemma 7; when Good does no happen with non-negligible probability, the distance is bounded by $2^{-\Omega(\lambda)}$ by Lemma 8.

Proof (Lemma 7). The first $\lambda/2$ $\mathsf{Execute}$ is perfectly simulated and we will show the second $\lambda/2$ $\mathsf{Execute}$ can also be simulated even only knowing one of z_0, z_1.

Without loss of generality, fixing $\mathbf{x}_{0,<}, \mathbf{x}_{1,<}, \mathbf{y}_{0,<}, \mathbf{y}_{1,<}$ sampled in the first $\lambda/2$ rounds, assume Good_0 happens and $\mathcal{A}$'s state is independent of s_0.The overall state of the system is,

$$\boldsymbol{\rho} \otimes \frac{1}{2}\left(|0\rangle\langle 0| + |1\rangle\langle 1|\right) \tag{1}$$

where $\boldsymbol{\rho}$ is the state of $\mathcal{A}$ and the register of s_1, the last register is for s_0.

In $\mathcal{F}^5_{z_0,z_1}$, the second half of the oracle access depends on $\mathbf{x}_{0,>}$, $\mathbf{x}_{1,>}$, $\mathbf{y}_{0,>}$, $\mathbf{y}_{1,>}$ in $\{0,1\}^{\lambda/2}$ and they are sampled in the following way: uniformly chooses $\mathbf{x}_{0,>}, \mathbf{x}_{1,>}, \mathbf{y}_{0,>}, \mathbf{y}_{1,>}$ such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle + s_0 = z_0$ and $\langle \mathbf{x}_{1,>}, \mathbf{y}_{1,>}\rangle + s_1 = z_1$. So with probability $1/2$, $\mathbf{x}_{0,>}, \mathbf{y}_{0,>}$ are sampled such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = z_0$ and with probability $1/2$, $\mathbf{x}_{0,>}, \mathbf{y}_{0,>}$ are sampled such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = \neg z_0$. The rest of the computation depends on $\mathbf{x}_{0,>}$, $\mathbf{x}_{1,>}$, $\mathbf{y}_{0,>}$, $\mathbf{y}_{1,>}$.

Sim_2 queries on z_1 and randomly samples $\hat{z}_0$. Because it knows both s_1 and z_1, it can generate $\mathbf{x}_{1,>}$ and $\mathbf{y}_{1,>}$ with exactly the same distribution in $\mathcal{F}^5_{z_0,z_1}$. However, Sim_2 does not know z_0 and it simply samples a random bit $\hat{z}_0$ uniformly at random together with $\mathbf{x}_{0,>}$ and $\mathbf{y}_{0,>}$ such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = \hat{z}_0$.

1. In $\mathcal{F}^5_{z_0,z_1}$:
   - With probability $1/2$, $\mathbf{x}_{0,>}$ and $\mathbf{y}_{0,>}$ are sampled uniformly at random such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = z_0$.
   - With probability $1/2$, $\mathbf{x}_{0,>}$ and $\mathbf{y}_{0,>}$ are sampled uniformly at random such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = \neg z_0$.
2. In Sim_2's simulation:
   - With probability $1/2$, $\mathbf{x}_{0,>}$ and $\mathbf{y}_{0,>}$ are sampled uniformly at random such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = 0$.
   - With probability $1/2$, $\mathbf{x}_{0,>}$ and $\mathbf{y}_{0,>}$ are sampled uniformly at random such that $\langle \mathbf{x}_{0,>}, \mathbf{y}_{0,>}\rangle = 1$.

Therefore it is easy to see that they are identical. □

Proof (Lemma 8). After the first $\lambda/2$ access to $\mathsf{Execute}$, the overall state of $\mathcal{A}$ and the ideal functionality registers (s_0, s_1), conditioned on no Good event happens, we have,

$$\begin{aligned} &p_{0,0} \cdot \boldsymbol{\rho}_{0,0} \otimes |00\rangle\langle 00| + p_{0,1} \cdot \boldsymbol{\rho}_{0,1} \otimes |01\rangle\langle 01| \\ &+ p_{1,0} \cdot \boldsymbol{\rho}_{1,0} \otimes |10\rangle\langle 10| + p_{1,1} \cdot \boldsymbol{\rho}_{1,1} \otimes |11\rangle\langle 11| \end{aligned}$$

where the last two registers are s_0 and s_1, $p_{b_0,b_1} = \Pr[s_0 = b_0,\ s_1 = b_1,\ \neg\mathsf{Good}]$. Also we have $p_{0,0} + p_{0,1} + p_{1,0} + p_{1,1} = \Pr[\neg\mathsf{Good}]$. The trace of the state may be smaller than 1 but to help prove the lemma, we *do not normalize* the state. We are going to show the following three claims. Lemma 9, 10 and 11 intuitively says that if no Good happens, $\mathcal{A}$ can not distinguish whether $s_0 = 0$ or 1, $s_1 = 0$ or 1 , $s_0 \oplus s_1 = 0$ or 1 respectively and therefore, $\boldsymbol{\rho}_{b_0,b_1}$ are pair-wise close to each other.

Lemma 9.

$$T\left(p_{0,0}\boldsymbol{\rho}_{0,0} + p_{0,1}\boldsymbol{\rho}_{0,1}, p_{1,0}\boldsymbol{\rho}_{1,0} + p_{1,1}\boldsymbol{\rho}_{1,1}\right) < 2^{-\lambda/20}$$

Lemma 10.

$$T\left(p_{0,0}\boldsymbol{\rho}_{0,0} + p_{1,0}\boldsymbol{\rho}_{1,0}, p_{0,1}\boldsymbol{\rho}_{0,1} + p_{1,1}\boldsymbol{\rho}_{1,1}\right) < 2^{-\lambda/20}$$

Lemma 11.

$$T\left(p_{0,0}\boldsymbol{\rho}_{0,0} + p_{1,1}\boldsymbol{\rho}_{1,1}, p_{0,1}\boldsymbol{\rho}_{0,1} + p_{1,0}\boldsymbol{\rho}_{1,0}\right) < 2^{-\lambda/20}$$

If they all holds, using triangle inequality, we can show that:

1. First, $p_{0,0}\boldsymbol{\rho}_{0,0}$ and $p_{1,1}\boldsymbol{\rho}_{1,1}$ are statistically close. From Lemma 9 and 10,

$$\begin{aligned} T(p_{0,0}\boldsymbol{\rho}_{0,0}, p_{1,1}\boldsymbol{\rho}_{1,1}) &= \frac{1}{2}\left|p_{0,0}\boldsymbol{\rho}_{0,0} - p_{1,1}\boldsymbol{\rho}_{1,1}\right|_1 \\ &\leq \frac{1}{2}\left|p_{0,0}\boldsymbol{\rho}_{0,0} + p_{0,1}\boldsymbol{\rho}_{0,1} - p_{1,0}\boldsymbol{\rho}_{1,0} - p_{1,1}\boldsymbol{\rho}_{1,1}\right|_1 \\ &\qquad + \frac{1}{2}\left|p_{0,0}\boldsymbol{\rho}_{0,0} - p_{0,1}\boldsymbol{\rho}_{0,1} + p_{1,0}\boldsymbol{\rho}_{1,0} - p_{1,1}\boldsymbol{\rho}_{1,1}\right|_1 \\ &\leq 2^{-\lambda/20+1} \end{aligned}$$

   It also gives us that $|p_{0,0} - p_{1,1}| < 2^{-\lambda/20+2}$. Since,

$$\begin{aligned} 2^{-\lambda/20+1} &\geq \frac{1}{2}\left|p_{0,0}\boldsymbol{\rho}_{0,0} - p_{1,1}\boldsymbol{\rho}_{1,1}\right|_1 \\ &\geq \frac{1}{2}\left|\left|p_{0,0}\boldsymbol{\rho}_{0,0}\right|_1 - \left|p_{1,1}\boldsymbol{\rho}_{1,1}\right|_1\right| \\ &= \frac{1}{2}\left|p_{0,0} - p_{1,1}\right|_1 \end{aligned}$$

2. Similarly to the first case, from Lemma 9 and 10,

$$T(p_{1,0}\boldsymbol{\rho}_{1,0}, p_{0,1}\boldsymbol{\rho}_{0,1}) \leq 2^{-\lambda/20+1} \text{ and } |p_{0,1} - p_{1,0}| \leq 2^{-\lambda/20+2}$$

3. Using the same argument as the first case, from Lemma 9 and 11, we have,

$$T(p_{0,0}\boldsymbol{\rho}_{0,0}, p_{0,1}\boldsymbol{\rho}_{0,1}) \leq 2^{-\lambda/20+1} \text{ and } |p_{0,0} - p_{0,1}| \leq 2^{-\lambda/20+2}$$

The probability of having no Good events is at least $2^{-c\lambda}$, in other words, $P = p_{0,0} + p_{0,1} + p_{1,0} + p_{1,1} > 2^{-c\lambda}$. Since p_{b_0,b_1} are pair-wise close, each p_{b_0,b_1} is at least $P/4 - 2^{-\Omega(\lambda)}$. We choose c such that $P/4 - 2^{-\Omega(\lambda)}$ is greater than 0.

We are going to show $\boldsymbol{\rho}_{b_0,b_1}$ are pair-wise statistically close. Let us take $\boldsymbol{\rho}_{0,0}$ and $\boldsymbol{\rho}_{1,1}$ as an example. We have,

$$\begin{aligned} \left|\boldsymbol{\rho}_{0,0} - \boldsymbol{\rho}_{1,1}\right|_1 &= \frac{1}{p_{0,0}}\left|p_{0,0}\boldsymbol{\rho}_{0,0} - p_{0,0}\boldsymbol{\rho}_{1,1}\right|_1 \\ &\leq \frac{1}{p_{0,0}}\left(\left|p_{0,0}\boldsymbol{\rho}_{0,0} - p_{1,1}\boldsymbol{\rho}_{1,1}\right|_1 + \left|(p_{0,0} - p_{1,1})\boldsymbol{\rho}_{1,1}\right|_1\right) \\ &\leq 2^{c\lambda} \cdot \left(2^{-\Omega(\lambda)} + 2^{-\Omega(\lambda)}\right) \\ &= 2^{-\Omega(\lambda)} \end{aligned}$$

By choosing c correctly (for example $c = 1/40 < 1/20$), we show $\boldsymbol{\rho}_{b_0,b_1}$ are pair-wise close.

Therefore the overall state is statistically close to $\Pr[\neg\mathsf{Good}]/4 \cdot \boldsymbol{\rho}_{0,0} \otimes \sum_{s_0} |s_0\rangle\langle s_0| \otimes \sum_{s_1} |s_1\rangle\langle s_1|$ conditioned on no Good happens. Since $\mathcal{A}$ is now statistically independent of both s_0, s_1, our simulator Sim_2 can perfectly simulate the rest of the $\mathsf{Execute}$, using the same argument as in the proof for Lemma 7. To complete the proof, we need to show Lemma 9, 10 and Lemma 11 holds.

Since Lemma 9, 10 are symmetric, we only give the proof for Lemma 9.

Proof (Lemma 9). Let $\boldsymbol{\rho}_0$ be the state of $\mathcal{A}$ conditioned on no Good event happens and $s_0 = 0$ (and taken partial trace over s_1) before making the next oracle query and p_0 be the probability of this case happens. Let $\boldsymbol{\rho}_1$ be the state of $\mathcal{A}$ conditioned on no Good event happens and $s_0 = 1$, and p_1 be the probability. We have that $p_0\boldsymbol{\rho}_0 = p_{0,0}\boldsymbol{\rho}_{0,0} + p_{0,1}\boldsymbol{\rho}_{0,1}$ and $p_1\boldsymbol{\rho}_1 = p_{1,0}\boldsymbol{\rho}_{1,0} + p_{1,1}\boldsymbol{\rho}_{1,1}$.

Let

$$\Delta\boldsymbol{\rho} = p_0\boldsymbol{\rho}_0 - p_1\boldsymbol{\rho}_1 = \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} \Delta_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} |b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'|$$

The trace distance of $p_0\boldsymbol{\rho}_0, p_1\boldsymbol{\rho}_1$ is defined as $\frac{1}{2} \cdot |\Delta\boldsymbol{\rho}|_1$. We will show that by making the next oracle query, $|\Delta\boldsymbol{\rho}|_1$ decreases by a constant factor. Thus, by making $\lambda/2$ queries, the trace distance becomes negligible. To simplify our proof, we partition $\Delta\boldsymbol{\rho}$ into the following 3×3 block matrix:

$$\Delta\boldsymbol{\rho} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix}$$

The first row contains all entries with $c = 0$, the second row contains all entries with $c = 1, b = 0$ and the third row contains all entries with $c = 1, b = 1$. Similarly, the first column contains all entries with $c' = 0$, the second column contains all entries with $c' = 1, b' = 0$ and the third column contains all entries with $c' = 1, b' = 1$. For example, F is in the second row, the third column, so it contains all entries with $c = c' = 1$ and $b = 0, b' = 1$,

$$F = \sum_{\mathsf{aux},\mathsf{aux}'} \Delta_{\substack{0,1,\mathsf{aux} \\ 1,1,\mathsf{aux}'}} |0, 1, \mathsf{aux}\rangle\langle 1, 1, \mathsf{aux}'| = H^\dagger$$

For a better presentation, we include the following two tables M_{bc} and $M_{(1-b)c}$. Each entry in both M_{bc} and $M_{(1-b)c}$ corresponds to a submatrix in $\Delta\boldsymbol{\rho}$. If the entry corresponding to a submatrix in M_{bc} is Y, it means all entries $|b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'|$ in the submatrix satisfy $bc = b'c'$; otherwise, they satisfy $bc \neq b'c'$. Similarly, if the entry corresponding to a submatrix in $M_{(1-b)c}$ is Y, it means all entries $|b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'|$ in the submatrix satisfy $(1-b)c = (1-b')c'$; otherwise, they satisfy $(1-b)c \neq (1-b')c'$.

$$\Delta\boldsymbol{\rho} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} \quad M_{bc} = \begin{bmatrix} Y & Y & N \\ Y & Y & N \\ N & N & Y \end{bmatrix} \quad M_{(1-b)c} = \begin{bmatrix} Y & N & Y \\ N & Y & N \\ Y & N & Y \end{bmatrix}$$

Claim. Define the matrix as above, we have,

$$\left|\begin{pmatrix} A & B \\ D & E \end{pmatrix}\right|_1 + |I|_1 = \left|\begin{pmatrix} A & B & \\ D & E & \\ & & I \end{pmatrix}\right|_1 \leq \left|\begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix}\right|_1$$

Proof. Define unitary U_{x_1} as $|b, c\rangle \to (-1)^{bcx_1}|b, c\rangle$. So we have

$$\begin{aligned}\left|\begin{pmatrix} A & B & \\ D & E & \\ & & I \end{pmatrix}\right| &= \left|\frac{1}{2}\sum_{x_1} U_{x_1} \Delta\boldsymbol{\rho} U_{x_1}^\dagger\right|_1 \\ &\leq \frac{1}{2}\sum_{x_1} \left|U_{x_1} \Delta\boldsymbol{\rho} U_{x_1}^\dagger\right|_1 = \frac{1}{2}\cdot 2 \cdot |\Delta\boldsymbol{\rho}|_1\end{aligned}$$

The first equality comes from the fact that s_0 is independent of x_1, so applying a random U_{x_1} is equivalent to zero out entries where $bc \neq b'c'$.

More formally, applying a unitary U_{x_1} for random x_1 on $\Delta\boldsymbol{\rho}$,

$$\begin{aligned}\frac{1}{2}\sum_{x_1} U_{x_1} \Delta\boldsymbol{\rho} U_{x_1}^\dagger &= \frac{1}{2}\sum_{x_1} \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} (-1)^{(bc+b'c')x_1} \Delta_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} |b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'| \\ &= \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} [bc = b'c'] \Delta_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} |b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'|\end{aligned}$$

From the above tables, for all entries in C, F, G, H, they do not satisfy $bc = b'c'$ and thus will be zeroed out. □

By making the next oracle query, we have 4 possible y_0, y_1 (we ignore subscripts or superscripts 'i'):

1. **Case 1:** $y_0 = 0, y_1 = 0$. In this case, define the resulting difference as $\Delta\boldsymbol{\rho}'_{0,0}$:

$$\begin{aligned}\Delta\boldsymbol{\rho}'_{0,0} &= |00\rangle\langle 00| \otimes \frac{1}{4} \cdot \sum_{x_0,x_1} U_{x_0,x_1} \Delta\boldsymbol{\rho} U_{x_0,x_1}^\dagger \\ &= |00\rangle\langle 00| \otimes \frac{1}{4} \cdot \sum_{x_0,x_1} \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} (-1)^{x_0((1-b)c+(1-b')c')+x_1(bc+b'c')} \\ &\qquad \cdot \Delta_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} |b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'|\end{aligned}$$

   The first two registers are y_0, y_1. Because y_0 is 0, the partial inner product s_0 does not change. Sine both x_0 and x_1 are free variables, it is equivalent to zero out all entries such that $bc \neq b'c'$ or $(1-b)c \neq (1-b')c'$. We have,

$$\Delta\boldsymbol{\rho}'_{0,0} = |00\rangle\langle 00| \otimes \begin{pmatrix} A & & \\ & E & \\ & & I \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{0,0}|_1 = |A|_1 + |E|_1 + |I|_1$.

2. **Case 2:** $y_0 = 0, y_1 = 1$. Because y_0 is 0, the partial inner product s_0 does not change. As discussed above, since both x_0, x_1 are free variables, it is equivalent to zero out all entries such that $bc \neq b'c'$ or $(1-b)c \neq (1-b')c'$. Moreover, $\mathcal{A}$ is not in event Good_0, we should also zero out all entries with $(1-b)c = 1$. The resulting difference is $\Delta\boldsymbol{\rho}'_{0,1}$:

$$\Delta\boldsymbol{\rho}'_{0,1} = |01\rangle\langle 01| \otimes \begin{pmatrix} A & \\ & I \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{0,1}|_1 = |A|_1 + |I|_1$.
3. **Case 3:** $y_0 = 1, y_1 = 0$. In this case, the new partial inner product s'_0 may flip depending on x_0. That is, if x_0 is 1, the new partial inner product becomes $s'_0 = \neg s_0$. Because $\Delta\rho$ is defined as $p_0\boldsymbol{\rho}_0 - p_1\boldsymbol{\rho}_1$, flipping s_0 introduces a term -1. Therefore, the term $(-1)^{x_0}$ will be included. Without conditioned on no Good_1 event happens, we have the difference is,

$$\begin{aligned}
&|10\rangle\langle 10| \otimes \frac{1}{4} \cdot \sum_{x_0,x_1} (-1)^{x_0} U_{x_0,x_1} \Delta\boldsymbol{\rho} U^\dagger_{x_0,x_1} \\
&= |10\rangle\langle 10| \otimes \frac{1}{4} \cdot \sum_{x_0,x_1} \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} (-1)^{x_0((1-b)c+(1-b')c'+1)+x_1(bc+b'c')} \\
&\qquad\qquad \cdot \Delta_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} |b,c,\mathsf{aux}\rangle\langle b',c',\mathsf{aux}'|
\end{aligned}$$

   So it is equivalent to zero out all entries such that $bc \neq b'c'$ or $(1-b)c = (1-b')c'$. Conditioned on no Good_1 event happens, in other words, $bc = b'c' = 0$, we have $\Delta\boldsymbol{\rho}'_{1,0}$ is

$$\Delta\boldsymbol{\rho}'_{1,0} = |10\rangle\langle 10| \otimes \begin{pmatrix} & B \\ D & \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{1,0}|_1 = |B|_1 + |D|_1$.
4. **Case 4:** Finally, for $y_0 = y_1 = 1$, the analysis is the same as Case 3 except we do not need to condition on $bc = b'c' = 0$. However, it turns out to be the same case, $\Delta\boldsymbol{\rho}'_{1,1}$ is

$$\Delta\boldsymbol{\rho}'_{1,1} = |11\rangle\langle 11| \otimes \begin{pmatrix} & B \\ D & \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{1,1}|_1 = |B|_1 + |D|_1$.

Therefore let $\Delta\boldsymbol{\rho}' = p'_0\boldsymbol{\rho}'_0 - p'_1\boldsymbol{\rho}'_1$ be the trace distance after making the oracle query. We have $\Delta\boldsymbol{\rho}' = \frac{1}{4}\sum_{y_0,y_1} \Delta\boldsymbol{\rho}'_{y_0,y_1}$. We want to show $|\Delta\boldsymbol{\rho}'|_1 \leq$

$\mathsf{constant} \cdot |\Delta\boldsymbol{\rho}|_1$. We already know

$$|\Delta\boldsymbol{\rho}'|_1 \leq \frac{1}{4}\sum_{y_0,y_1} |\Delta\boldsymbol{\rho}'_{y_0,y_1}|_1 = \frac{2|A|_1 + 2|B|_1 + 2|D|_1 + |E|_1 + 2|I|_1}{4}.$$

Next, we need the following lemma from [BK90]:

Lemma 12 (Restate of main theorem in [BK90]). *For every matrix $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times m}$, $D \in \mathbb{C}^{m\times n}$, $E \in \mathbb{C}^{m\times m}$, let M be the block matrix $M = \begin{pmatrix} A & B \\ D & E \end{pmatrix}$. We have $|A|_1^2 + |B|_1^2 + |D|_1^2 + |E|_1^2 \leq |M|_1^2$.*

Combining with Lemma 12 and Cauchy-Schwarz inequality, we have,

$$\begin{aligned}
|\Delta\boldsymbol{\rho}'|_1 &\leq \frac{2|A|_1 + 2|B|_1 + 2|D|_1 + |E|_1 + 2|I|_1}{4} \\
&= \frac{1}{2}|A|_1 + \frac{1}{2}|B|_1 + \frac{1}{2}|D|_1 + \frac{1}{4}|E|_1 + \frac{1}{2}|I|_1 \\
&\leq \sqrt{(3\times(1/2^2) + (1/4)^2)\left(|A|_1^2 + |B|_1^2 + |D|_1^2 + |E|_1^2\right)} + \frac{1}{2}|I|_1 \\
&\leq \frac{\sqrt{13}}{4}\left|\begin{pmatrix} A & B \\ D & E \end{pmatrix}\right|_1 + \frac{1}{2}|I|_1 \\
&\leq \frac{\sqrt{13}}{4}|\Delta\boldsymbol{\rho}|_1
\end{aligned}$$

Let $\Delta\boldsymbol{\rho}^*$ be the difference after $\lambda/2$ rounds. We have $|\Delta\boldsymbol{\rho}^*|_1 \leq 0.91^{\lambda/2} < 2^{-\lambda/20}$. □

Proof (Lemma 11). The proof is very similar to the proof for Lemma 9, but the case by case analysis is different and gives a better bound. Let $\boldsymbol{\rho}_0$ be the state of $\mathcal{A}$ conditioned on no Good event happens and $s_0 \oplus s_1 = 0$ before making the next oracle query and p_0 be the probability of this case happens. Let $\boldsymbol{\rho}_1$ be the state of $\mathcal{A}$ conditioned on no Good event happens and $s_0 \oplus s_1 = 1$, and p_1 be the probability. We have that $p_0\boldsymbol{\rho}_0 = p_{0,0}\boldsymbol{\rho}_{0,0} + p_{1,1}\boldsymbol{\rho}_{1,1}$ and $p_1\boldsymbol{\rho}_1 = p_{0,1}\boldsymbol{\rho}_{0,1} + p_{1,0}\boldsymbol{\rho}_{1,0}$. Define

$$\Delta\boldsymbol{\rho} = p_0\boldsymbol{\rho}_0 - p_1\boldsymbol{\rho}_1 = \sum_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} \Delta_{\substack{b,c,\mathsf{aux} \\ b',c',\mathsf{aux}'}} |b,c,\mathsf{aux}\rangle\langle b',c',\mathsf{aux}'|$$

The trace distance of $\boldsymbol{\rho}_0, \boldsymbol{\rho}_1$ is defined as $\frac{1}{2}\cdot|\Delta\boldsymbol{\rho}|_1$. We partition $\Delta\boldsymbol{\rho}$ into the following 3×3 block matrix:

$$\Delta\boldsymbol{\rho} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix}$$

By making the next oracle query, we have 4 possible y_0, y_1:

1. **Case 1:** $y_0 = 0, y_1 = 0$.
   In this case, we have a term $(-1)^{(bc+b'c')x_1+((1-b)c+(1-b')c')x_0}$ for each entry $|b, c, \mathsf{aux}\rangle\langle b', c', \mathsf{aux}'|$. It is equivalent to zero out all entries such that $bc \neq b'c'$ or $(1-b)c \neq (1-b')c'$. The resulting difference is $\Delta\boldsymbol{\rho}'_{0,0}$:

$$\Delta\boldsymbol{\rho}'_{0,0} = |00\rangle\langle 00| \otimes \begin{pmatrix} A & & \\ & E & \\ & & I \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{0,0}|_1 = |A|_1 + |E|_1 + |I|_1$.
2. **Case 2:** $y_0 = 0, y_1 = 1$ (and Good_0 does not happen).
   In this case, we have a term $(-1)^{(bc+b'c'+1)x_1+((1-b)c+(1-b')c')x_0}$ for each entry. It is equivalent to zero out all entries such that 1) $bc = b'c'$ or 2) $(1-b)c \neq (1-b')c'$, 3) $(1-b)c = (1-b')c' = 1$. The resulting difference is $\Delta\boldsymbol{\rho}'_{0,1}$:

$$\Delta\boldsymbol{\rho}'_{0,1} = |01\rangle\langle 01| \otimes \begin{pmatrix} & & C \\ & & \\ G & & \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{0,1}|_1 = |C|_1 + |G|_1$.
3. **Case 3:** $y_0 = 1, y_1 = 0$ (and Good_1 does not happen).
   In this case, we have a term $(-1)^{(bc+b'c')x_1+((1-b)c+(1-b')c'+1)x_0}$ for each entry. It is equivalent to zero out all entries such that 1) $bc \neq b'c'$ or 2) $(1-b)c = (1-b')c'$, 3) $bc = b'c' = 1$. The resulting difference is $\Delta\boldsymbol{\rho}'_{1,0}$:

$$\Delta\boldsymbol{\rho}'_{1,0} = |10\rangle\langle 10| \otimes \begin{pmatrix} & B & \\ D & & \\ & & \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{1,0}|_1 = |B|_1 + |D|_1$.
4. **Case 4:** Finally, for $y_0 = y_1 = 1$.
   In this case, we have a term $(-1)^{(bc+b'c'+1)x_1+((1-b)c+(1-b')c'+1)x_0}$ for each entry. It is equivalent to zero out all entries such that 1) $bc = b'c'$ or 2) $(1-b)c = (1-b')c'$. The resulting difference is $\Delta\boldsymbol{\rho}'_{1,1}$:

$$\Delta\boldsymbol{\rho}'_{1,1} = |11\rangle\langle 11| \otimes \begin{pmatrix} & & F \\ & H & \end{pmatrix}$$

   We know that $|\Delta\boldsymbol{\rho}'_{1,1}|_1 = |F|_1 + |H|_1$.

So let $\Delta\boldsymbol{\rho}' = p'_0\boldsymbol{\rho}'_0 - p'_1\boldsymbol{\rho}'_1$ be the trace distance after making the oracle query. We have $\Delta\boldsymbol{\rho}' = \frac{1}{4}\sum_{y_0,y_1} \Delta\boldsymbol{\rho}'_{y_0,y_1}$. We want to show $|\Delta\boldsymbol{\rho}'|_1 \leq \mathsf{constant} \cdot |\Delta\boldsymbol{\rho}|_1$.

By Cauchy-Schwarz inequality and Lemma 12, we have

$$\begin{aligned}
|\Delta\boldsymbol{\rho}'|_1 &\leq \frac{1}{4}\sum_{y_0,y_1}|\Delta\boldsymbol{\rho}'_{y_0,y_1}|_1 \\
&= \frac{|A|_1+|B|_1+|C|_1+|D|_1+|E|_1+|F|_1+|G|_1+|H|_1+|I|_1}{4} \\
&\leq \sqrt{\frac{9}{16}\left(|A|_1^2+|B|_1^2+|C|_1^2+|D|_1^2+|E|_1^2+|F|_1^2+|G|_1^2+|H|_1^2+|I|_1^2\right)} \\
&\leq \frac{3}{4}\cdot|\Delta\boldsymbol{\rho}|_1
\end{aligned}$$

Let $\Delta\boldsymbol{\rho}^*$ be the difference after $\lambda/2$ rounds. We have $|\Delta\boldsymbol{\rho}^*|_1 \leq 0.75^{\lambda/2} < 2^{-\lambda/20}$. □

As we finishes the proof for Lemma 9, 10 and 11, we show as long as the probability of having no Good event is at least $2^{-c\lambda}$, the algorithm is statistically independent of both s_0 and s_1. Therefore Sim_2 notices there is no Good event happens. Sim_2 simply samples the rest vectors $\mathbf{x}_{0,>}, \mathbf{x}_{1,>}, \mathbf{y}_{0,>}, \mathbf{y}_{0,>}$ conditioned on uniformly random $\hat{z}_0, \hat{z}_1$. This completes the proof for Lemma 8. □

Finally, we are ready to complete the proof for Theorem 1.

Proof. For every z_0, z_1 and every quantum algorithm $\mathcal{A}$, let $\mathcal{B} = \mathsf{Sim}_0(\mathcal{A})$, by Lemma 1 we have

$$T\left(\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\mathcal{B} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right)\right) = 0$$

where $\mathcal{B}$ has the following properties: (1) every (j, tag) pair will be queried at most once; (2) every $1 \leq j \leq \lambda$, (j, x) is always queried before (j, y).

Let $\mathcal{C} = \mathsf{Sim}_1(\mathcal{B})$, by Lemma 2, we have,

$$T\left(\left(\mathcal{B} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\mathcal{C} \Longleftrightarrow \mathcal{F}^4_{z_0,z_1}\right)\right) = 0$$

By Lemma 3, we have,

$$T\left(\left(\mathcal{C} \Longleftrightarrow \mathcal{F}^4_{z_0,z_1}\right), \left(\mathcal{C} \Longleftrightarrow \mathcal{F}^5_{z_0,z_1}\right)\right) < 2^{-\Omega(\lambda)}$$

Let $\mathcal{D} = \mathsf{Sim}_2(\mathcal{C})$, by Lemma 5, we have,

$$T\left(\left(\mathcal{C} \Longleftrightarrow \mathcal{F}^5_{z_0,z_1}\right), \left(\mathcal{D} \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{z_0,z_1}\right)\right) < 2^{-\Omega(\lambda)}$$

Combining the above together, let $\widetilde{\mathsf{Sim}} = \mathsf{Sim}_2 \circ \mathsf{Sim}_1 \circ \mathsf{Sim}_0$, for every z_0, z_1, every quantum algorithm $\mathcal{A}$,

$$T\left(\left(\mathcal{A} \Longleftrightarrow \mathcal{F}^*_{z_0,z_1}\right), \left(\widetilde{\mathsf{Sim}}(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{z_0,z_1}\right)\right) < 2^{-\Omega(\lambda)}$$

□

3.2 c-OTM construction for multiple inputs

The Scheme. For $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$, the sender chooses $4n$ uniformly random strings $\mathbf{x}_0^i, \mathbf{y}_0^i, \mathbf{x}_1^i, \mathbf{y}_1^i \in \{0,1\}^\lambda$ such that for all i, b, $\mathbf{z}_{b,i} = \langle \mathbf{x}_b^i, \mathbf{y}_b^i \rangle$. The sender prepares $2n \cdot \lambda$ q-OTM instances of the following input pairs $(x_{0,j}^i, x_{1,j}^i)$ and $(y_{0,j}^i, y_{1,j}^i)$ for $1 \leq i \leq n, 1 \leq j \leq \lambda$. The functionality of the construction is equivalent to the following (Functionality 7).

Functionality 7 The construction for strings $\mathcal{F}^*_{\mathbf{z}_0,\mathbf{z}_1}$

1: Create **:** Upon inputs $(\mathbf{z}_0, \mathbf{z}_1)$ where both are bit strings of length n,
   1. The sender chooses $4n$ random strings $\mathbf{x}_0^i, \mathbf{y}_0^i, \mathbf{x}_1^i, \mathbf{y}_1^i \in \{0,1\}^\lambda$ such that for all i, b, $z_{b,i} = \langle \mathbf{x}_b^i, \mathbf{y}_b^i \rangle$.
   2. The sender stores pairs $(x_{0,j}^i, x_{1,j}^i)$ and $(y_{0,j}^i, y_{1,j}^i)$ for $1 \leq i \leq n, 1 \leq j \leq \lambda$. It also prepares a table $\{m_{x,j}^i = 0\}$ and $\{m_{y,j}^i = 0\}$ indicating if the j-th input pair corresponding to either $\mathbf{x}$ or $\mathbf{y}$ has been queried or not.

2: Execute **:** Upon a quantum state $\sum_{b,c} \alpha_{b,c} |b,c\rangle$ and classical inputs i, j, tag from the receiver ($\mathsf{tag} \in \{x, y\}$), let us assume $\mathsf{tag} = x$. The same holds for $\mathsf{tag} = y$.
   1. The sender first checks if $m_{x,j}^i = 0$. If not, the information about $x_{0,j}^i$ and $x_{1,j}^i$ has already been queried by the receiver and has been deleted by the sender. The sender simply does nothing and sends the state back.
   2. Otherwise, the sender applies the unitary $U_i : |b,c\rangle \to (-1)^{c \cdot x_{b,j}^i} |b,c\rangle$. It then sends the quantum state back to the receiver, marks $m_{x,j}^i$ as 1 and deletes both $x_{0,j}^i$ and $x_{1,j}^i$.

We have n parallel instances of c-OTM for each bit pair $z_{0,i}, z_{1,i}$.

Definition 4. *Let $\mathcal{F}_1 \cdots \mathcal{F}_n$ be n ideal functionality with* Create *and* Execute*. Define the notation $\mathcal{A} \Longleftrightarrow (\mathcal{F}_1, \cdots, \mathcal{F}_n)$ as the density matrix of $\mathcal{A}$ in the following interaction: each $\mathcal{F}_i$ initializes with* Create *at the beginning, and $\mathcal{A}$ can access to $\mathcal{F}_i$.*Execute *for all i.*

Therefore, $\mathcal{A}$'s state in the interaction with $\mathcal{F}^*_{\mathbf{z}_0,\mathbf{z}_1}$ can be denoted as

$$\mathcal{A} \Longleftrightarrow (\mathcal{F}^*_{z_{0,1},z_{1,1}}, \cdots, \mathcal{F}^*_{z_{0,n},z_{1,n}})$$

Theorem 2. *The above construction is a c-OTM in the q-OTM model. In other words, there exists an efficient simulator* $\mathsf{Sim}_{\mathsf{multi}}$*, for all $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$, and all quantum algorithm $\mathcal{A}$,*

$$\mathsf{Tr}\left(\left(\mathcal{A} \Longleftrightarrow (\mathcal{F}^*_{z_{0,1},z_{1,1}}, \cdots, \mathcal{F}^*_{z_{0,n},z_{1,n}})\right), \left(\mathsf{Sim}_{\mathsf{multi}}(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{\mathsf{cOTM}}_{\mathbf{z}_0,\mathbf{z}_1}\right)\right) = n2^{-\Omega(\lambda)}$$

Proof. We have the following claims. The proofs are identical to those in the proof for Theorem 1.

Claim. We can assume every (i, j, tag) pair will only be queried at most once.

Claim. We can assume for every $1 \le i \le n, 1 \le j \le \lambda$, (i, j, x) is always queried before (i, j, y). In other words, if 1) both are queried, then (i, j, x) is queried before (i, j, y); 2) only one if queried, then it is (i, j, x); 3) neither is queried.

Similar to Corollary 1, we have,

Corollary 2. *There exists an efficient simulator* $\mathsf{Sim}_{\mathsf{multi},1}$*, for every fixed* $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$*, for every quantum algorithm* $\mathcal{A}$*,*

$$T\big((\mathcal{A} \Longleftrightarrow (\mathcal{F}^*_{z_{0,1},z_{1,1}} \cdots \mathcal{F}^*_{z_{0,n},z_{1,n}})), \\ (\mathsf{Sim}_{\mathsf{multi},1}(\mathcal{A}) \Longleftrightarrow (\mathcal{F}^4_{z_{0,1},z_{1,1}}, \cdots, \mathcal{F}^4_{z_{0,n},z_{1,n}}))\big) = 0$$

Moreover, for each i*,* $\mathsf{Sim}_{\mathsf{multi},1}(\mathcal{A})$ *queries* $\mathcal{F}^4_{z_{0,i},z_{0,i}}.\mathsf{Execute}$ *on inputs* j *in the order from* 1 *to* λ*.*

Proof. The proof is identical to that for Lemma 2/Corollary 1. The simulator $\mathsf{Sim}_{\mathsf{multi},1}$ runs $\widetilde{\mathsf{Sim}}_1$ and simulates for each $\mathcal{F}^*_{z_{0,j},z_{1,j}}$ independently. □

Corollary 3. *For every* $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$*, for every quantum algorithm* $\mathcal{A}$ *that queries (for each* i*)* $\mathcal{F}^4_{z_{0,i},z_{0,i}}.\mathsf{Execute}$ *on inputs* j *in the order from* 1 *to* λ*,*

$$T\big((\mathcal{A} \Longleftrightarrow (\mathcal{F}^4_{z_{0,1},z_{1,1}} \cdots \mathcal{F}^4_{z_{0,n},z_{1,n}})), \\ (\mathcal{A} \Longleftrightarrow (\mathcal{F}^5_{z_{0,1},z_{1,1}}, \cdots, \mathcal{F}^5_{z_{0,n},z_{1,n}}))\big) < n2^{-\Omega(\lambda)}$$

Proof. The difference of $\mathcal{F}^4$ and $\mathcal{F}^5$ is how $\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1$ are sampled. We have shown the statistical distance is at most $2^{-\Omega(\lambda)}$ in Lemma 3. Therefore for n parallel instances, it is bounded by $n2^{-\Omega(\lambda)}$. □

Corollary 4. *There exists an efficient simulator* $\mathsf{Sim}_{\mathsf{multi},2}$*, for every fixed* $\mathbf{z}_0, \mathbf{z}_1 \in \{0,1\}^n$*, for every quantum algorithm* $\mathcal{A}$ *which (for every* i*) queries* $\mathcal{F}^5_{z_{0,i},z_{0,i}}.\mathsf{Execute}$ *on inputs* j *in the order from* 1 *to* λ*,*

$$T\big((\mathcal{A} \Longleftrightarrow (\mathcal{F}^5_{z_{0,1},z_{1,1}} \cdots \mathcal{F}^5_{z_{0,n},z_{1,n}})), \\ (\mathsf{Sim}_{\mathsf{multi},2}(\mathcal{A}) \Longleftrightarrow (\mathcal{F}^{\mathsf{cOTM}}_{z_{0,1},z_{1,1}}, \cdots, \mathcal{F}^{\mathsf{cOTM}}_{z_{0,n},z_{1,n}}))\big) < n2^{-\Omega(\lambda)}$$

Proof. We define the following hybrids, in **Hybrid** h, we define $\mathsf{Sim}_{\mathrm{Hyb},h}$:

1. The ideal functionality it interacts with is

$$\mathcal{F}^{(h)}_{\mathbf{z}_0,\mathbf{z}_1} = (\mathcal{F}^{\mathsf{cOTM}}_{z_{0,1},z_{1,1}}, \cdots, \mathcal{F}^{\mathsf{cOTM}}_{z_{0,h},z_{1,h}}, \mathcal{F}^5_{z_{0,h+1},z_{1,h+1}}, \cdots, \mathcal{F}^5_{z_{0,n},z_{1,n}})$$

   The first h instances are replaced with c-OTM ideal functionality.
2. $\mathsf{Sim}_{\mathrm{Hyb},h}$ runs $\mathcal{A}$ as a subroutine.
3. For $i \le h$, for queries $(|\phi\rangle, i, j)$ to the i-th instance, $\mathsf{Sim}_{\mathrm{Hyb},h}$ acts like Sim_2 in Lemma 5: if Good event ever happens, it knows $\mathcal{A}$ loses the information about $z_{b,i}$ for exactly one b; otherwise, $\mathcal{A}$ loses information about both $z_{0,i}$ and $z_{1,i}$. Therefore $\mathsf{Sim}_{\mathrm{Hyb},h}$ can simulate all queries to the i-the ideal functionality by just accessing to $\mathcal{F}^{\mathsf{cOTM}}_{z_{0,i},z_{1,i}}$.

4. For all $i > h$, the simulator simply forwards queries from $\mathcal{A}$ to the i-th idea functionality.

Finally we have the following corollary, from Lemma 5.

Corollary 5. *For every fixed $z_0, z_1 \in \{0,1\}^n$, for every quantum algorithm $\mathcal{A}$ which (for every i) queries $\mathcal{F}^5_{z_{0,i},z_{0,i}}$.*Execute *on inputs j in the order from 1 to λ,, for every $0 \leq h < n$,*

$$T\left(\mathsf{Sim}_{Hyb,h}(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{(h)}_{\mathbf{z}_0,\mathbf{z}_1}, \mathsf{Sim}_{Hyb,h+1}(\mathcal{A}) \Longleftrightarrow \mathcal{F}^{(h+1)}_{\mathbf{z}_0,\mathbf{z}_1}\right) < 2^{-\Omega(\lambda)}$$

Proof. The proof follows from that of Lemma 5. □

By triangle inequality, we complete the proof of Theorem 2. □

References

Aru19. Frank et al. Arute. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. 1

ATTU16. Mayuresh Vivekanand Anand, Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Post-quantum security of the cbc, cfb, ofb, ctr, and xts modes of operation. In *Post-Quantum Cryptography*, pages 44–63. Springer, 2016. 1, 9

BGS13. Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs - (extended abstract). In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 344–360. Springer, Heidelberg, August 2013. 2, 4, 11

BGZ18. Anne Broadbent, Sevag Gharibian, and Hong-Sheng Zhou. Towards quantum one-time memories from stateless hardware. Cryptology ePrint Archive, Report 2018/960, 2018. https://eprint.iacr.org/2018/960. 2, 11

BK90. Rajendra Bhatia and Fuad Kittaneh. Norm inequalities for partitioned operators and an application. *Mathematische Annalen*, 287(1):719–726, 1990. 27

BZ13a. Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 592–608. Springer, Heidelberg, May 2013. 1, 5, 9

BZ13b. Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Heidelberg, August 2013. 1, 9

CGS08. Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 545–562. Springer, Heidelberg, April 2008. 2

DFNS14. Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. Superposition attacks on cryptographic protocols. In Carles Padró, editor, *ICITS 13*, volume 8317 of *LNCS*, pages 142–161. Springer, Heidelberg, 2014. 1, 9

DJ92. David Deutsch and Richard Jozsa. Rapid solutions of problems by quantum computation. In *Proceedings of the Royal Society of London A*, 1992. 3, 5, 35

ECKM20. Ehsan Ebrahimi, Céline Chevalier, Marc Kaplan, and Michele Minelli. Superposition attack on ot protocols. Cryptology ePrint Archive, Report 2020/798, 2020. https://eprint.iacr.org/2020/798. 9

GHS16. Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 60–89. Springer, Heidelberg, August 2016. 1, 9

GIS+10. Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 308–326. Springer, Heidelberg, February 2010. 2, 4

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2008. 2

GYZ17. Sumegha Garg, Henry Yuen, and Mark Zhandry. New security notions and feasibility results for authentication of quantum data. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 342–371. Springer, Heidelberg, August 2017. 1, 9

Kat07. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, Heidelberg, May 2007. 2

KLLN16. Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 207–237. Springer, Heidelberg, August 2016. 1, 9

KM10. Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2682–2685, 2010. 1, 9

LC97. Hoi-Kwong Lo and Hoi Fung Chau. Is quantum bit commitment really possible? *Physical Review Letters*, 78(17):3410, 1997. 9

May97. Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Physical review letters*, 78(17):3414, 1997. 9

MS08. Tal Moran and Gil Segev. David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 527–544. Springer, Heidelberg, April 2008. 2

Nay99. Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *40th FOCS*, pages 369–377. IEEE Computer Society Press, October 1999. 9

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. 1

SS17. Thomas Santoli and Christian Schaffner. Using simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Information & Computation*, 17:65–78, 2017. 1, 9

Zha12. Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012. 1, 9

Zha16. Mark Zhandry. A note on quantum-secure PRPs. Cryptology ePrint Archive, Report 2016/1076, 2016. http://eprint.iacr.org/2016/1076. 1, 9

Zha19. Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2019, Part II*, LNCS, pages 239–268. Springer, Heidelberg, August 2019. 9

A Technical Proofs

A.1 Proof for Lemma 3

Proof. We prove the case for $z = 0$, the proof is similar for $z = 1$.

Let us look at the distribution of the first method. This is a uniform distribution over all valid $(\mathbf{x}, \mathbf{y})$ pairs such that $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. So we need to know the total number of $\langle \mathbf{x}, \mathbf{y} \rangle$ pairs such that $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Let us say the number is $Q_0(\lambda)$ and the number of pairs with inner product equals to 1 is $Q_1(\lambda)$. First we have $Q_0(\lambda) + Q_1(\lambda) = 4^\lambda = 2^{2\lambda}$.

Assume $\mathbf{y} \neq \mathbf{0}$, then there are equal number of $\mathbf{x}$ such makes the inner product 0 or 1. Because without loss of generality $y_1 = 1$, for every $\mathbf{x}$ such that $\langle \mathbf{x}, \mathbf{y} \rangle = 1$, we can simply flip x_1 and flip the inner product. If $\mathbf{y} = \mathbf{0}$, the inner product is always 0. Therefore we have $Q_0(\lambda) - Q_1(\lambda) = 2^\lambda$ which gives us $Q_0(\lambda) = (2^{2\lambda} + 2^\lambda)/2$ and $Q_1(\lambda) = (2^{2\lambda} - 2^\lambda)/2$.

Therefore for the first method, the distribution is a uniform distribution over valid $(\mathbf{x}, \mathbf{y})$ pairs, and the probability is $2/(2^{2\lambda} + 2^\lambda)$.

For the second method, let $\mathbf{x}' = \mathbf{x}'_< || \mathbf{x}'_>$ and $\mathbf{y}' = \mathbf{y}'_< || \mathbf{y}'_>$. Since $\mathbf{x}'_<$ and $\mathbf{y}'_<$ are sampled uniform at random, there are $Q_0(\lambda/2)$ pairs of $\mathbf{x}'_<$ and $\mathbf{y}'_<$ with partial inner product 0 and $Q_1(\lambda/2)$ pairs with partial inner product 1. Each of these pair has probability $1/4^{\lambda/2} = 1/2^\lambda$.

If the partial inner product is 0, the inner product of $\mathbf{x}'_>$ and $\mathbf{y}'_>$ should be 0 as well. This goes to our discussion for the first method and it is a uniform distribution, with probability $1/Q_0(\lambda/2)$. If the partial inner product is 1, the distribution of $\mathbf{x}'_>$ and $\mathbf{y}'_>$ is a uniform distribution over all pairs of inner product 1, and each probability is $1/Q_1(\lambda/2)$.

Therefore by definition of Δ, we have,

$$\begin{aligned} 2 \cdot \Delta\left((\mathbf{X}, \mathbf{Y}), (\mathbf{X}', \mathbf{Y}')\right) &= Q_0(\lambda/2)^2 \cdot \left|1/Q_0(\lambda) - 1/2^\lambda \cdot 1/Q_0(\lambda/2)\right| \\ &\quad + Q_1(\lambda/2)^2 \cdot \left|1/Q_0(\lambda) - 1/2^\lambda \cdot 1/Q_1(\lambda/2)\right| \\ &= O(2^{-\lambda/2}) \end{aligned}$$

Because one can verify both $Q_0(\lambda/2)^2, Q_1(\lambda/2)^2$ are of order $O(2^{2\lambda})$ and both differences are of order $O(2^{-5/2 \cdot \lambda})$. □

B Quantum-accessible Oblivious Transfer

In this section, we briefly discuss extending our results to the oblivious transfer setting. 1-out-of-2 oblivious transfer has essentially the same functionality as a one-time memory, just with the functionality resulting from an interactive protocol as opposed to a piece of hardware. our main result easily extends to the oblivious transfer setting. This means any secure "quantum-accessible" OT - where the ideal functionality lets the receiver query on superposition - can be lifted to a secure "classical-accessible" OT.

We leave building secure "quantum-accessible" OT in the standard model as an open question. Here, we consider the usual OT reorientation protocol that converts random OT correlations into an OT scheme. We show that, if the receiver can make quantum queries to the sender, the protocol is an information-theoretically secure "quantum-accessible" OT. We note that the reorientation protocol is *not* "classical-accessible", as the Deutsch-Jozsa [DJ92] attack applies to learn the parity of the two bits. By applying our compiler, however, we are able to achieve an information-theoretically secure "classical-accessible" OT from random OT correlations, despite the receiver's ability to query the sender on quantum superpositions.

B.1 From OT Correlations to Quantum-accessible OT

The scheme. We consider an information-theoretic setting where the sender and receiver begin with random OT correlations. Suppose the ideal resource has given the sender two random bits x_0, x_1, and the receiver c, x_c for a random bit c.

Now, consider the following OT protocol:

- The receiver, on input b, sends the message $d = b \oplus c$.
- The sender, on inputs m_0, m_1 and the message d, sends $(x_d \oplus m_0, x_{d\oplus 1} \oplus m_1)$.

Theorem 3. *The above construction is a quantum-accessible OT scheme.*

Proof. Receiver's security follows straightforward because c is completely random and hidden from the sender's view.

To show it has sender's security against a quantum query, we need to show a superposition query by the receiver $\mathcal{R}$ can be simulated by a superposition query made to $U_{x_0,x_1} : |b, y\rangle \to (-1)^{y \cdot m_b}|b, y\rangle$ and therefore its ideal functionality is the same with that defined in Definition 2.

Our overall proof will consist of two steps:

- First, we note that due to the randomness of the scheme, in particular the bit $x_{c\oplus 1}$ that are not available to the receiver, the view of the adversary after the queries is a mixed state. We will demonstrate an equivalent operation on the adversary's state that results in the same mixed state. This alternative operation involves a partial measurement of the adversary's state — which importantly is independent of the messages m_0, m_1 — as well as a phase

term. At this point, the simulated view still requires knowing both m_0 and m_1 to generate the phase. Note that the trick we use is similar to that in the proof of Theorem 1.
– Next, we will show how to simulate the phase term in step 1, using the results of the measurement in step 1, as well as a single quantum query to the quantum ideal functionality.

The receiver's state right before the query consists 3 qubits $|d, y_0, y_1\rangle$, d for the query to the OT instance, and $|y_0, y_1\rangle$ for the responses. The receiver also keeps some auxiliary state z, which includes c, x_c, as well as any intermediate computations.

Lemma 13. *The following two operations on the adversary's state form equivalent final states:*

– *Answer the query as in the scheme.*
– *First, measure y_{d+c+1}. Then perform the unitary $U_{m_0,m_1,c}$ where*

$$U_{m_0,m_1,c}|d, y_0, y_1, z\rangle = (-1)^{y_{d+c}x_c+\sum_b y_b m_b}|d, y_0, y_1, z\rangle$$

We use + for addition mod 2.

Proof (Lemma 13). Fixing x_c, c, denote the state of the receiver $\mathcal{S}'$ before making the query as a mixed state $\boldsymbol{\rho}$:

$$\boldsymbol{\rho} = \sum_{x_{c+1}} \sum_{\substack{d,y_0,y_1,z \\ d',y_0',y_1',z'}} \rho^{c,x_c}_{d,y_0,y_1,z,d',y_0',y_1',z'} |d, y_0, y_1, z\rangle\langle d', y_0', y_1', z'|$$

We now consider what happens when the adversary's query is answered as in the scheme. The entry $|d, y_0, y_1, z\rangle\langle d', y_0', y_1', z'|$ is multiplied by a phase

$$\begin{aligned}&(-1)^{y_0(x_d+m_0)+y_1(x_{d+1}+m_1)+y_0'(x_{d'}+m_0)+y_1'(x_{d'+1}+m_1)}\\ =&(-1)^{\sum_b\left((y_b'+y_b)m_b+y_b x_{d+b}+y_b' x_{d'+b}\right)}\end{aligned}$$

Because $\sum_b y_b x_{d+b} = \sum_b y_{b+c+d} x_{c+b}$, we can rewrite the phase as the following (note that b is not 'b' in our OT scheme, instead it is just a subscript of the summation),

$$(-1)^{\sum_b\left((y_b'+y_b)m_b+(y_{d+c+b}'+y_{d+c+b})x_{c+b}\right)}$$

Since x_{c+1} is not given to $\mathcal{S}'$, we can trace out x_{c+1} by averaging. This means the entry $|d, y_0, y_1, z\rangle\langle d', y_0', y_1', z'|$ is multiplied by the following phase,

$$\frac{1}{2}\sum_{x_{c+1}}(-1)^{\sum_b\left((y_b'+y_b)m_b+(y_{d+c+b}'+y_{d+c+b})x_{c+b}\right)}$$

We now evaluate this term. Notice that this sum is 0 unless $y_{d+c+1} = y_{d'+c+1}'$ since x_{c+1} is a free variable. Therefore we can zero out all entries whose $y_{d+c+1} \neq$

$y'_{d'+c+1}$, as discussed in the preliminary, it is equivalent to measure the value of y_{d+c+1}. Thus, $|d, y_0, y_1, z\rangle\langle d', y'_0, y'_1, z'|$ is multiplied by

$$(-1)^{\sum_b \left((y'_b+y_b)m_b+(y'_{d+c+b}+y_{d+c+b})x_{c+b}\right)} \cdot \delta(y_{d+c+1}, y'_{d'+c+1})$$

Therefore, by (1) first performing a partial measurement on the value of y_{d+c+1} and (2) then performing the unitary $U_{m_0,m_1,c}$, each entry $|d, y_0, y_1, z\rangle\langle d', y'_0, y'_1, z'|$ is multiplied by the same phase above. This completes the proof of Lemma 13. □

We are almost ready to describe our simulator. The simulator can clearly measure the adversary's state, since this is independent of the sender's message m_0, m_1. The goal is then to implement $U_{m_0,m_1,c}$ while only making a single quantum query to the ideal functionality.

Lemma 14. *Given the result of measuring y_{d+c+1}, it is possible to implement $U_{m_0,m_1,c}$ by making a single query to the ideal functionality.*

Proof (Lemma 14). To simplify the notation, we will switch to considering a distribution over pure states rather than a mixed state; it is then straightforward to generalize to mixed states, which are simply convex combinations of pure states.

Fixing c, x_c, assume we measure the value of y_{d+c+1} and get r. The state collapses to

$$|\psi_r\rangle = \sum_{\substack{d,y_0,y_1,z \\ y_{d+c+1}=r}} \alpha_{d,y_0,y_1,z}|d, y_0, y_1, r\rangle$$

Our goals is to turn this into the state $|\phi_r\rangle = U_{m_0,m_1,r}|\psi_r\rangle$,

$$|\phi_r\rangle = \sum_{\substack{d,y_0,y_1,z \\ y_{d+c+1}=r}} (-1)^{\sum_b y_b m_b + y_{d+c}x_c}\, \alpha_{d,y_0,y_1,z}|d, y_0, y_1, r\rangle$$

Since $y_{d+c+1} = r$, the summation $\sum_b y_b m_b$ is equal to $y_{d+c}m_{d+c} + rm_{d+c+1}$. Recall that an overall phase factor does not change a quantum state, we will therefore multiply the state $|\psi_r\rangle$ with a global phase $(-1)^{r(m_0+m_1)}$ which is also equal to $(-1)^{r(m_{d+c}+m_{d+c+1})}$. Notice that multiplying this phase cancels out $(-1)^{rm_{d+c+1}}$ and replaces it with $(-1)^{rm_{d+c}}$.

Therefore, getting $|\phi_r\rangle$ is equivalent to the following,

$$|\phi'_r\rangle = \sum_{\substack{d,y_0,y_1,z \\ y_{d+c+1}=r}} (-1)^{(y_{d+c}+r)m_{d+c}+y_{d+c}x_c}\, \alpha_{d,y_0,y_1,z}|d, y_0, y_1, r\rangle$$

To get $|\phi'_r\rangle$, the simulator can just do the following steps:

1. Map $|d, y_0, y_1, z\rangle$ to $|d, y_0, y_1, z\rangle \otimes |d + c, y_{d+c} + r\rangle$.

2. Send the query $|d+c, y_{d+c}+r\rangle$ to the ideal functionality. This will introduce a phase factor of $(-1)^{m_{d+c}(y_{d+c}+r)}$.
3. Uncompute $|d+c, y_{d+c}+r\rangle$.
4. Apply a simple phase unitary that maps $|d, y_0, y_1, z\rangle$ to $(-1)^{y_{d+c}x_c}|d, y_0, y_1, z\rangle$.

The composition of these 4 steps maps $|d, y_0, y_1, z\rangle$ to $(-1)^{(y_{d+c}+r)m_{d+c}+y_{d+c}x_c}|d, y_0, y_1, r\rangle$. Thus it maps $|\psi_r\rangle$ to $|\phi'_r\rangle$ as desired. □

Putting everything together, our simulator measures y_{d+c+1} register and then simulate the phase term by Lemma 14. □