A Quantum Approach for Reducing Communications in Classical Cryptographic Primitives

Jiayu Zhang*[1]

[1]Zhongguancun Laboratory

September 29, 2023

Abstract

How could quantum cryptography help us achieve what are not achievable in classical cryptography? In this work we consider the following problem, which we call *succinct RSPV for classical functions* (sRCF). Suppose f is a function described by a polynomial time classical Turing machine, which is public; the client would like to sample a random x as the function input and use a protocol to send $f(x)$ to the server. What's more, (1) when the server is malicious, what it knows in the passing space should be no more than $f(x)$; (2) the communication should be succinct (that is, independent to the running time of evaluating f). Solving this problem in classical cryptography seems to require strong cryptographic primitives.

We show that, perhaps surprisingly, it's possible to solve this problem with quantum techniques under much weaker assumptions. By allowing for quantum communication and computations, we give a protocol for this problem assuming only collapsing hash functions [26]. Our work conveys an interesting message that quantum cryptography could outperform classical cryptography in a new type of problems, that is, to reduce communications in meaningful primitives without using heavy classical cryptographic primitives.

1 Introduction

The development of quantum information science has significant impacts in cryptography. One field that is growing rapidly is quantum cryptography, which makes use of quantum techniques to achieve cryptographic tasks. Famous examples include quantum key distribution [20], multiparty quantum computation [2], unclonable cryptography [23], etc.

One remarkable feature about quantum cryptography is it allows us to go beyond classical cryptography in various problems [20, 3, 13]. For example, quantum key distribution [20] achieves information-theoretic secure key exchange, which is not possible classically. Discovering new quantum advantage in this field is of great theoretical interest on its own, and could also lead to deeper understandings and new protocols for related problems.

*zhangjy@zgclab.edu.cn

Size of Communication in Classical Cryptography In this work we care about the size of communication of cryptographic primitives. In various problems, it is possible to make the communication *succinct*, which means, the size of total communications is independent to the problem size, and only depends on the security parameter [15]. One famous example is the Kilian's succinct argument protocol for verifying NP [16]. Despite the rapid growing of quantum cryptography, the following problem remains unexplored:

Could quantum cryptography help to reduce communications in classical cryptographic tasks?

1.1 Our Works, Part I: the Problem and the Basic Construction

In this work we formalize a classical cryptographic problem called *succinct RSPV for classical functions* (sRCF). We give a quantum cryptographic protocol for this problem that in a sense outperforms classical cryptographic solutions.

1.1.1 Succinct RSPV for classical functions (sRCF)

The problem that we consider is as follows. Consider a client and a server, and a function f described by a polynomial time classical Turing machine, and an input length n; f, n are publicly available. The client would like to sample an input $x \leftarrow_r \{0,1\}^n$ randomly and sends $f(x)$ to the server. We would like to design a protocol for this problem with roughly the following correctness, security and efficiency requirement:

- (Correctness) When both parties follow the protocol, the protocol passes successfully and the client gets a random $x \in \{0,1\}^n$ and the server gets the corresponding $f(x)$.
- (Security) For any polynomial time malicious server, there is a server-side simulator taking $f(x)$ as its input such that: in the case that the protocol passes successfully, the joint state of the client and the simulator's output should be approximately indistinguishable to the final state of executing the protocol. This intuitively means what the server knows is no more than $f(x)$.
- (Efficiency) The honest behaviors of both parties run in polynomial time; what's more, the total size of the communication is independent to the running time of evaluating f.

The RSPV in its name comes from quantum cryptography [12, 11, 28]. RSPV in existing works aims at preparing quantum states on the server-side, with similar correctness and security requirements as described above; here we consider RSPV for classical functions,[1] and what makes it nontrivial is that the communication is succinct.

This notion encapsulates many existing notions in classical cryptography as its special cases. As an example, consider $f(k) = \mathrm{PRF}_k(1)||\mathrm{PRF}_k(2)||\cdots||\mathrm{PRF}_k(N)$ for some large N. To solve the sRCF problem for this f in the world of classical cryptography, the typical approach[2] is by constrained PRF [19], which requires relatively stronger assumptions like the Learning-with-Errors assumption [22]; solving the problem for general f seems to require much heavier assumptions like the indistinguishability obfuscation [1, 14].

[1] Here we abuse the notion a little bit since RSPV in existing works typically works on a fixed state family while here we consider a function encoded by a Turing machine.

[2] (Based on a discussion with a classical cryptography expert.)

Background: cryptographic assumptions In this work we care about strength of assumptions. Indistinguishability obfuscation is considered to be a very strong assumption, for which there is still no post-quantum secure construction from well-founded assumptions. The Learning-with-Errors (LWE) assumption is typically considered to be reasonably good, but still much stronger than assumptions like collision-resistant or collapsing hash functions [26, 25].

1.1.2 Protocol constructions

By making use of quantum techniques, we give a protocol for the sRCF problem assuming collapsing hash functions.

Our protocol goes as follows:

Protocol 1. *Suppose the function is f and the input length is n.*

1. *The client samples bit strings $x_0, x_1 \leftarrow_r \{0,1\}^n$, $x_0^{(outpad)}, x_1^{(outpad)} \leftarrow_r \{0,1\}^n$ and prepares and sends*
$$\frac{1}{\sqrt{2}}(|x_0\rangle|x_0^{(outpad)}\rangle + |x_1\rangle|x_1^{(outpad)}\rangle) \tag{1}$$
to the server.

2. *The server evaluates f in superposition and gets*
$$\frac{1}{\sqrt{2}}(|x_0\rangle|x_0^{(outpad)}\rangle|f(x_0)\rangle + |x_1\rangle|x_1^{(outpad)}\rangle|f(x_1)\rangle) \tag{2}$$
Let's name the three quantum registers in (2) as input register, output pad register and output register correspondingly. Then the server does bitwise Hadamard measurement on the input register and gets measurement outcome $d^{(inp)} \in \{0,1\}^n$ and the remaining state
$$\frac{1}{\sqrt{2}}((-1)^{d^{(inp)}\cdot x_0}|x_0^{(outpad)}\rangle|f(x_0)\rangle + (-1)^{d^{(inp)}\cdot x_1}|x_1^{(outpad)}\rangle|f(x_1)\rangle) \tag{3}$$
The server sends back $d^{(inp)}$ to the client. The client rejects if $d^{(inp)}$ is all-zero (which only happens with negligible probability).

3. *The client chooses roundtype to be* test *or* comp *by flipping coins.* test *is used to test the server's behavior and* comp *is to generate the target output. The two branches of the protocol are designed as follows:*
   - *roundtype =* test*:*
     (a) *The client asks the server to measure the output pad register on Hadamard basis; the server performs the operation and sends back the outcome $d^{(outpad)} \in \{0,1\}^n$.*
     The server measures all the output register in (3) on the Hadamard basis. Suppose the measurement outcome is $d^{(out)}$, then the following condition should be satisfied:
$$d^{(inp)}\cdot(x_0\oplus x_1)+d^{(outpad)}\cdot(x_0^{(outpad)}\oplus x_1^{(outpad)})+d^{(out)}\cdot(f(x_0)\oplus f(x_1)) \equiv 0 \mod 2 \tag{4}$$
     Note that $d^{(out)}$ might be long so it doesn't work to send $d^{(out)}$ back directly to the client for testing. The following steps are to verify the equation (4) within succinct communication.

(b) *Suppose h is a collapsing hash function. The server computes* $c = h(d^{(out)})$ *and sends back c to the client.*

(c) *The client sends* $(x_0^{(outpad)}, x_1^{(outpad)})$ *to the server.*

(d) *The client sends* (x_0, x_1) *to the server.*

(e) *The client and the server perform a succinct argument of knowledge for the following NP relation, where* $d^{(out)}$ *is the witness:*

$$d^{(out)} \textit{ satisfies } (4) \textit{ and } h(d^{(out)}) = c \tag{5}$$

- *roundtype* = comp*:*

  The client asks the server to measure all the bits in (3) *on the standard basis and report the output pad register outcome, denoted as* r*.*

  The client checks $r \in \{x_0^{(outpad)}, x_1^{(outpad)}\}$ *and stores the corresponding* x *and disgards the other.*

Note that states in the form of (1) and Hadamard test on key pair superpositions have already been used in a series of existing works [8, 17, 29].

We could first give an intuitive explanation of this protocol. The client first asks the server to do Hadamard measurement on the input register, which in the honest setting collapses the state to (3). (3) almost achieves what we want since collapsing it to the standard basis leads to a state where the server only knows $f(x)$. The danger is, the server might cheat: for example, the server might keep x_0, x_1 in the input register and only return some garbage; the server might be able to get some information other than $f(x)$ before x is destroyed by Hadamard measurement. How could we rule out this type of attack? We make use of the Hadamard test: one observation is, if the client asks the server to make Hadamard measurement on all the remaining qubits, the joint Hadamard test condition would certify that the server has performed what it should do (a deeper explanation will be given in Section 1.2). Furthermore, this process need to be made succinct, which leads to the step 3.b∼3.e of the protocol.

1.1.3 Function family

In our work, we aim at function families that satisfy a very general property called *inner-product-uniform*: Suppose $f = (f_n)_{n\in\mathbb{N}}$, each f_n is a function of $\{0,1\}^n \to \{0,1\}^m$ for some $m = m(n)$. This function is called inner-product-uniform if for any $(d^{(inp)}, d^{(out)}) \in \{0,1\}^n \times \{0,1\}^m$ that is not all-zero,

$$\Pr_{x\leftarrow_r\{0,1\}^n}[x \cdot d^{(inp)} + f(x) \cdot d^{(out)} \equiv 0 \mod 2] \approx_{\mathsf{negl}(n)} \frac{1}{2}$$

that is, the inner product of $(x, f(x))$ and any non-zero $(n+m)$-dimension vector is approximately equally possible to be odd or even. This is satisfied by various common functions like the PRF expansion function discussed in Section 1.1.1.

1.1.4 A more explicit treatment of security definition

The security of sRCF is defined by simulation, as the treatment of RSPV in [6, 12, 28].

Recall the goal of the protocol is to build the following joint state between the client and the server:

$$x \leftarrow_r \{0,1\}^n, \text{ client holds } x, \text{ server holds } f(x) \tag{6}$$

Then the security statement is roughly as follows. Suppose f is an inner-product-uniform function family, ϵ is a constant, we claim our protocol has the following property. For any polynomial time quantum adversary, there exists a simulator Sim working solely on the server side of (6) such that, if the protocol passes with probability $\geq 1-\delta$:

$$\text{on the passing space, final state of the protocol} \approx^{ind}_{\epsilon} \mathsf{Sim}(\text{equation (6)}) \text{ for some } \epsilon = \mathsf{poly}(\delta) \quad (7)$$

where the $\approx^{ind}_{\epsilon}$ notation (formalized in Notation 2.3) denotes approximate indistinguishability of the left and right sides of (7). Note here the distinguisher has access to both the client and the server.

Remark on the security dependency on ϵ Note that here we fix ϵ to be a constant (instead of a function of n). In this case the problem is already nontrivial. When we consider the more general case where $\epsilon = \epsilon(n)$ our protocol incurs a very undesirable blowup. In our current security proof, the approximation error in (7) is actually $\epsilon + 2^{\mathsf{poly}(1/\epsilon)}\mathsf{negl}(n)$. We will discuss this issue in more detail in Section 1.2.2, 4.3.

Finally the protocol and security statement described above is slightly different from the notion described in Section 1.1.1 in the sense that in the statement above δ is considered to be small, and we would like to make $1-\delta$ small so that the failing case could be merged into the approximation error. This could be achieved by the RSPV amplification procedure described in [30].

1.2 Our Works, Part II: Security Proof

The security proof turns out to be nontrivial and requires technical innovations. We give an overview of the security proof here and formalize the proof in Section 4.

The first step is to analyze the sAoK step in step 3.e. Thanks to existing works [10], we already have an post-quantum analysis of sAoK for NP relation. Applying its knowledge extraction property, we could roughly say, by the end of step 3.d, the server holds a string $d^{(out)}$ (possibly in superposition) that satisfies (5). But how could the existence of $d^{(out)}$ imply the existence of simulator in (7)?

One way to understand the security proof here is to consider simplified variants of our protocol and see how we could prove what we want. Although proofs in the simplified case do not immediately work in the general case, these discussions provide and illustrate new techniques that will be used in the full proof.

1.2.1 Simplified protocol for illustrating properties of the Hadamard test

Let's first consider the following simplified variant of Protocol 1.

Toy Protocol 1. *1.2. Step 1 and step 2 are the same as Protocol 1.*

3. • *If roundtype* $=$ test*:*

  (a) The client asks the server to measure the output pad register on the Hadamard basis and send back the outcome $d^{(outpad)} \in \{0,1\}^n$*.*
  The client asks the server to measure all the output register on the Hadamard basis. The server sends back the measurement outcome $d^{(out)}$*.*
  The client checks (4)*.*

Let's introduce the following states for analyzing the protocol:

- Denote the two branches in (1) as $|\varphi_0^1\rangle, |\varphi_1^1\rangle$ (so $|\varphi_0^1\rangle + |\varphi_1^1\rangle = (1)$).
- Then define $|\varphi_b^2\rangle$ as the final joint state of taking the initial state to be $|\varphi_b^1\rangle$ and running the protocol to the end of step 2.
- Then define $|\varphi_b^3\rangle$ as the final joint state after the simplified protocol 1 above is completed.

Making use of the properties of Hadamard test, by a little bit of calculations we know passing the checking of (4) implies

$$\Pi_{\mathsf{pass}} |\varphi_0^3\rangle \approx \Pi_{\mathsf{pass}} |\varphi_1^3\rangle$$
$$\Pi_{\mathsf{fail}} |\varphi_0^3\rangle \approx -\Pi_{\mathsf{fail}} |\varphi_1^3\rangle$$

Note that the determination of pass and fail is based on (4), which only requires the following information:

$$d^{(inp)} \cdot (x_0 \oplus x_1) + d^{(outpad)} \cdot (x_0^{(outpad)} \oplus x_1^{(outpad)}) \mod 2, f(x_0), f(x_1) \tag{8}$$

Then the following operation, denoted by O, maps $|\varphi_0^3\rangle$ to $|\varphi_1^3\rangle$ given the information in (8):

1. Use the information in (8) to determine whether a basis d is in the Π_{pass} space or in the Π_{fail} space; flip the sign of the Π_{fail} space.

We could even further reduce the information that the operation needs: notice that the first term in (8) is only a 1-bit information that helps to decide whether it's in the passing space or failing space; even if this is determined arbitrarily, the outcome is just a global (-1) phase that is undetectable when these d, x are classical. More explicitly, we can define O' that only makes use of the following information:

$$x_0, f(x_1) \tag{9}$$

And $O' |\varphi_0^3\rangle$ is approximately indistinguishable to $|\varphi_1^3\rangle$.

A further statement claims $|\varphi_0^2\rangle$ could be transformed to $|\varphi_1^2\rangle$ with the following information:

$$x_0, f(x_1), x_0^{(outpad)}, x_1^{(outpad)} \tag{10}$$

This intuitively achieves what we want: the important thing here is what the server knows about x_1 is no more than $f(x_1)$; knowing x_0 does not give the server any information about x_1.

The key difference of the simplified protocol above from our full protocol is in the full protocol, the server first sends back $c = h(d^{(out)})$. Then before the sAoK part, the client has to send out (x_0, x_1) to allow the server to run the sAoK. In other words, the server actually gets information that goes beyond (8) (which even includes x_1 itself) and the argument above breaks down. How could we handle this problem?

1.2.2 Simplified protocol 2 and a key lemma about hash preimage extraction

We consider another simplified variant of the protocol that captures how the hash commitment step takes effect. The collapsing hash function [26] is a quantum generalization of collision-resistant hash function.

Toy Protocol 2. *1. An efficient adversary prepares a state in $\boldsymbol{D} \otimes \boldsymbol{c}$, which should intuitively satisfy $h(d) = c$ where $d \in \{0,1\}^m$ is the value of $\boldsymbol{D}$ and c is the value of $\boldsymbol{c}$.*

Send c to the client.

2. *The client samples* $x \in_r \{0,1\}^m$*, sends it to the server and asks for* d*.*
The passing condition is $x \cdot d \mod 2 = 0$ *and* $h(d) = c$ *is satisfied.*

Note that the server does not have access to x initially and x is only sampled out in the second step. Thus even in the honest setting the passing probability is $\frac{1}{2}$.

The $\frac{1}{2}$ passing probability is obviously optimal once d is fixed. But what we want is in a sense its inverse version, that is,

If the server could pass with probability p, it holds a d such that $h(d) = c$ with probability roughly $2p$.

We will formalize a lemma that captures this intuition and give the proof in Section 4.3. Intuitively what the proof does is to sample a series of x. If we focus on a specific x, d could be extracted from the server's state with probability p. Combining these different sampling of x and extractions allows us to extract d with higher probability, which could be close to $2p$.

We note that, possibly due to limitations of our current techniques, the extraction of d in our current proof (Lemma 4.3 in Section 4.3) has an undesirable exponential blowup that is related to p. This is not a problem if p is a fixed constant, which leads to constant approximation error in (7). A more desirable statement is to make the protocol work for $\epsilon = 1/\mathsf{poly}(n)$.

1.2.3 An overview of the full proof

What we already have is, by the knowledge extraction property of the sAoK protocol, the server holds $d^{(out)}$ that satisfies the conditions by the end of step 3.d of the test roundtype. What we want to argue is, before the beginning of step 3.d, the server already knows such a $d^{(out)}$. Our proof, on the high level, goes roughly as follows:

1. Prove the server knows (more formally, there exists an efficient extractor that could extract) $d^{(out)}$ if only one of $\{x_0, x_1\}$ (say, x_0) is sent to the server.
2. Prove the server already knows $d^{(out)}$ before the client sends $\{x_0, x_1\}$.

Once we complete the argument above, the problem is roughly reduced to the simplified setting described in Section 1.2.1.

Furthermore, the difficulties in two steps above are actually the same. Thus we could zoom in to the first step: how to remove the reliance on x_1 on the server-side extraction of $d^{(out)}$.

Note that in each step of the protocol, the state appearing in the protocol could be written as the sum of two states: one state is the output state when the state prepared in step 1 of Protocol 1 is $\frac{1}{\sqrt{2}} |x_0\rangle |x_0^{(outpad)}\rangle$, the other state is the output state when the state prepared in step 1 is $\frac{1}{\sqrt{2}} |x_1\rangle |x_1^{(outpad)}\rangle$. We temporarily call these two states the x_0-branch and the x_1-branch. Then the construction of the extraction of $d^{(out)}$ is further divided into two steps.

1. Show that there exists a server-side extraction algorithm with access to x_0 that extracts $d^{(out)}$ on the x_0-branch.
2. Show that the extraction operation above also works in the x_1-branch; what's more, the output of the extraction algorithm on these two branches are coherent in the right way.

The first step is roughly what we have done in the Section 1.2.2: before x_1 is sent out, the server-side of the x_0 branch does not depend on x_1. Thus sending out x_1 is like the "sampling x" operation in Toy Protocol 2.

The nontrivial part is how to move from one branch to the other branch. Here we make use of the collapsing property of hash functions together with the Hadamard test conditions to achieve this leap.

1. To move from one branch to the other, a key step here is to return to consider the behavior of the protocol in the normal execution, that is, the input state is the sum of the two branches. The passing condition says, the server's probability of holding a $d^{(out)}$ that falls into this subspace is small:

$$\text{The Hadamard test condition (equation (4)) is not satisfied,}$$
$$\text{but the hash checking } (h(d^{(out)}) = c) \text{ is satisfied.} \tag{11}$$

   Here we need to be careful that the server's operation here is different from the "extraction of $d^{(out)}$" we discussed just now. But we could still say something by relating them to each other. Aided with the collapsing property of the hash function, we could show that:

   For any efficient operation (like the extraction of $d^{(out)}$) operating on the sum of two branches, it's hard to output a string in subspace (11).

2. Now we return to the extraction of $d^{(out)}$ that works on a single branch. Recall that we could extract $d^{(out)}$ from the $\boldsymbol{x}_0$ branch and we want to prove the extraction also works directly on the $\boldsymbol{x}_1$ branch. Now the argument just now tells us:

   If we compare the output of the extraction operation on $\boldsymbol{x}_0$ branch and the $\boldsymbol{x}_1$ branch, their components in subspace (11) cancel with each other (approximately sum to zero).

   But it does not directly allow us to argue about the relation of the states outside subspace (11). Here we make use of the properties of Hadamard test and the fact that the function we are testing satisfies the "inner-product-uniform" property. We could show:

   The square norm of the state on subspace (11) is approximately half of the whole state.

3. Now we are ready to leap from one branch to the other. What we know is (1) two branches cancel with each other on the subspace (11); (2) the square norm of the branch on this subspace is half of the whole. By linear algebra we could prove the components of two branches on the complement of subspace (11) are approximately equal — otherwise their sum will not be big enough to make its norm equal to 1.

The argument above omits a lot of technical details: for example, everything above is only approximately correct, but netherless we could bound all the error terms by small values. The full proof is given in Section 4 and we give a step-by-step bottom-up overview there.

1.3 Related Works

Studies of succinctness in quantum cryptography There has been several works that study succinctness in quantum cryptography, like [29, 4]. [29] studies succinctness of quantum communication in blind quantum computation problem; [4] studies succinct communication in classical verification of quantum computation problem. As far as we know, the advantage of quantum cryptography in getting succinct communication in classical cryptographic primitives has not been discovered before.

Classical primitives that share similarities to our problem There are a series of classical primitives that share some similarity to our notion including garbling [27], succinct randomized encoding [7], laconic function evaluation [21], etc. As far as we know, our notion is different from these existing notions: we allow interactions so our notion is different from garbling and succinct randomized encoding; the communication size in our problem should be independent to the output size of f so it's different from laconic function evaluation. Despite this, considering the extensive development of classical cryptography, it's not a surprise that our notion has already been considered in existing works of classical cryptography.

Quantum works that share technical similarities to our construction Our construction contains a sense of revocation or deletion, which is a quantum phenomenon. A series of existing works also make use of this phenomenon to achieve nontrivial cryptographic tasks, for example, [24, 9, 5, 28]. However, our usage of this quantum phenomenon is different from these existing techniques so we consider our technique to be in the same class as but parallel to these existing works.

1.4 Summary and Discussion

The big message of this work is that we study and discover a new type of quantum advantage in cryptography.

Our result also leads to several open questions. For example:

- There is an undesirable exponential blowup related to the approximation error (see Section 1.1.4, 1.2.2).
- Our construction is not necessarily optimal in terms of number of rounds, assumptions, or the type of functions that it could support.
- Another question that seems open is to give formal evidences that the problem that we consider is hard to construct in the classical world.

We also hope our construction could be useful in other problems, or could inspire the discovery of more quantum advantage in other problems.

2 Preliminaries

We refer to [15] for basics of cryptography and [18] for basics of quantum information. In this section we clarify notations and review important notions.

Notation 2.1. The registers are denoted as bold font, for example, $\boldsymbol{x}, \boldsymbol{d}$. The values in these registers are denoted with the corresponding normal font, x, d.

Notation 2.2. For a superoperator $\mathcal{E}$ we use $\mathcal{E}(\cdot)$ to denote its operation on an input state. We also use this notation for unitaries and projections.

Note that we also use $C(w)$ to denote running a classical circuit C on an input w.

Notation 2.3. Denote a single-bit projection onto value 0 as Π_0. For state family $\rho = (\rho_n)_{n\in\mathbb{N}}$, $\sigma = (\sigma_n)_{n\in\mathbb{N}}$, we write $\rho \approx^{ind:\mathcal{F}}_{\epsilon} \sigma$ if for any algorithm $D \in \mathcal{F}$ there is $\mathrm{tr}(\Pi_0(D(\rho))) \approx_\epsilon \mathrm{tr}(\Pi_0(D(\sigma)))$.
We write $\rho \approx^{ind}_{\epsilon} \sigma$ if in the definition above $\mathcal{F}$ is the set of polynomial time quantum algorithm.
We write $\rho \approx^{ind} \sigma$ if $\rho \approx^{ind}_{\mathsf{negl}(n)} \sigma$.

Notation 2.4. On a quantum state ρ, notation $\Pr[\rho \to v]$ is defined by $\mathrm{tr}(\Pi_v(\rho))$ where Π_v is the projection onto the space of v. Both notations are used.

Below we introduce a series of notions and notations for hash functions.

Notation 2.5. We use $\mathrm{range}(h)$ to denote the range set of a hash function.

Definition 2.1 (Collision-resistant hash functions)**.** A function family $(h_n)_{n\in\mathbb{N}}$ is called post-quantum collision-resistant hash functions if for any polynomial time adversary Adv,

$$\Pr[\mathsf{Adv}(1^n) \to (x_0, x_1), x_0, x_1 \in \{0,1\}^n, h(x_0) = h(x_1)] = \mathsf{negl}(n)$$

Definition 2.2 (Collapsing hash functions [26])**.** A function family $(h_n)_{n\in\mathbb{N}}$ is called collapsing if the following property is satisfied. For any polynomial-time quantum adversary that takes 1^n as input and outputs a quantum state ρ in $\boldsymbol{W} \otimes \boldsymbol{X} \otimes \boldsymbol{C}$, define the following operation:

- $\Pi_{h(\boldsymbol{X})=\boldsymbol{C}}$ as the projection onto the space that the values in register $(\boldsymbol{X}, \boldsymbol{C})$, denoted by (x, c), satisfies $x \in \{0,1\}^n, h(x) = c$.
- $\mathsf{COPY}_{\boldsymbol{X}\to\boldsymbol{aux}}$ is an operation that copies the value of $\boldsymbol{X}$ to an empty register $\boldsymbol{aux}$.
- $\mathcal{F}$ is the set of polyonmial time operations that has full access to $\boldsymbol{W}$ and only classical access to $\boldsymbol{C}$ and does not have access to $\boldsymbol{aux}$.

Then:

$$\mathsf{COPY}_{\boldsymbol{X}\to\boldsymbol{aux}}(\Pi_{h(\boldsymbol{X})=\boldsymbol{C}}(\rho)) \approx^{ind:\mathcal{F}} \Pi_{h(\boldsymbol{X})=\boldsymbol{C}}(\rho)$$

Theorem 2.1 ([26])**.** *If a hash function is collapsing, then it's post-quantum collision-resistant.*

3 Succinct RSPV for Classical Functions

In this section we formalize the notion of succinct RSPV for classical functions (sRCF), and give the protocol and the theorems.

3.1 Definitions

As the preparation we first clarify our setting of cryptographic protocols.

Definition 3.1. In our work, the protocol works between a client and a server. Furthermore, there are a series of message registers that stores classical messages between them (quantum states are transmitted directly). These message registers are read-only once a message is stored in them.

The operations in a protocol could be in the following types:

- The client does its local computations.
- The client sends messages or states to the server. Messages are stored in the message register while states are sent to an empty server-side register directly.
- The server does its local computations; note that the server only has read-only access to the message registers.
- The server sends back messages to the client (note that we only consider classical messages for this type. That is, in this type of operations, the server writes its computation result to an empty message register).

Finally the client decides whether to accept to reject. (That is, the client writes the flag of decision in a transcript register.)

Below we formalize the notion of sRCF.

Definition 3.2 (Setting of sRCF)**.** An sRCF protocol is defined for a function family $f = (f_n)_{n\in\mathbb{N}}$, $f_n : \{0,1\}^n \to \{0,1\}^m$ where $m = m(n)$ is the output length. In the protocol there is a client and a server, and the protocol takes an input size n as its input; initially the description of a polynomial time classical Turing machine T that computes f is publicly available to both parties.

Definition 3.3 (Completeness of sRCF)**.** An sRCF protocol for f is complete with completeness error c if in the honest setting in the end of the protocol the joint state of the client and the server is c-close to the following: for input length n, the client gets $x \leftarrow_r \{0,1\}^n$, and the server gets the corresponding $f(x)$.

Definition 3.4 (Efficiency of sRCF)**.** In an sRCF protocol the honest behavior of both the client and the server should be in polynomial time and the communication should be in $\mathsf{poly}(n)$, independent to the running time of evaluating f.

As a preparation of sRCF soundness definition, we introduce the following notation from [28].

Notation 3.1. For a function f and input length n as defined in Definition 3.2, define the joint state between the client and the server

$$\rho_{tar} = \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}} \underbrace{|x\rangle}_{\text{client}} \otimes \underbrace{|f(x)\rangle}_{\text{server}}$$

Below we make the input size n implicit.

Below we give two slightly different definitions for the soundness of sRCF. The first definition contains a soundness parameter s and an approximation error ϵ. The second definition only has a parameter ϵ. We will use the first definition as an intermediate definition in the construction later and consider the second definition as the final target.

Definition 3.5 (Soundness of sRCF)**.** An sRCF protocol for f has soundness s and approximation error ϵ if for any polynomial time adversary Adv there exists an efficiently computable server-side simulator $\mathsf{Sim}^{\mathsf{Adv}}$ such that one of the following two is true:

- (Small passing probability)
$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}}(1^n))) < s$$
- (Simulation-based security)
$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}}(1^n)) \approx_{\epsilon}^{ind} \Pi_{\mathsf{pass}}(\mathsf{Sim}^{\mathsf{Adv}}(\rho_{tar}))$$
where the distinguisher has classical access to the client and transcript and full access to the server.

Definition 3.6 (Soundness of sRCF)**.** An sRCF protocol for f is sound with approximation error ϵ if for any polynomial time adversary Adv there exists an efficiently computable server-side simulator $\mathsf{Sim}^{\mathsf{Adv}}$ such that
$$\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}}(1^n)) \approx_{\epsilon}^{ind} \Pi_{\mathsf{pass}}(\mathsf{Sim}^{\mathsf{Adv}}(\rho_{tar}))$$
where the distinguisher has classical access to the client and transcript and full access to the server.

The time complexity and communication complexity are formalized as the usual definitions.

3.2 Protocol Design

We formalize our protocol in this section.

3.2.1 Preparation: succinct argument of knowledge for NP

As a preparation, we review the notion and existing results for succinct argument of knowledge (succinct AoK, sAoK) for NP relation [16].

Definition 3.7 (Setting and efficiency of sAoK)**.** A succinct argument of knowledge (sAoK) protocol takes a security parameter κ and a polynomial-size classical circuit C as inputs. Initially the honest server holds a witness w such that $C(w) = 0$.

Its efficiency requirements are: (1) the honest behaviors of both parties run in polynomial time; (2) the communication size is $\mathsf{poly}(\kappa)$.

The intuitive goal of sAoK is to convince the client that the server really holds a witness w such that $C(w) = 0$. This is formalized by the following completeness and knowledge extractability conditions:

Definition 3.8 (AoK completeness)**.** We say an sAoK protocol π_{sAoK} is complete if: when the server is honest and initially holds w such that $C(w) = 0$, the client outputs pass in the end with probability $1 - \mathsf{negl}(\kappa)$.

Definition 3.9 (Post-quantum AoK knowledge extractability)**.** We say an sAoK protocol π_{sAoK} has knowledge extraction parameter (s, s_k) if the following statement holds for the protocol:

For any polynomial time quantum adversary Adv and adversarial initial state ρ_{init}, there exists a polynomial time computable $\mathsf{Ext}^{\mathsf{Adv}}$ such that:

If:

$$\Pr[\pi_{sAoK}^{\mathsf{Adv}}(\rho_{init}, 1^\kappa) \to \mathsf{pass}] \geq s$$

then:

$$\Pr[C(\mathsf{Ext}^{\mathsf{Adv}}(\rho_{init})) = 0] \geq s_k \tag{12}$$

By Kilian's protocol [16] and its post-quantum security proof [10], we have the following theorem for its existence:

Theorem 3.1. *Assuming collapsing hash functions, there exists an sAoK protocol for NP relation that is complete, efficient and has knowledge extraction parameter* $(s, s' - \mathsf{negl}(\kappa))$ *for any constant* $s, s' \in (0,1), s' < s$.

3.2.2 sRCF protocol

Our sRCF protocol is as follows. This protocol is a more explicit repetition of Protocol 1.

Protocol 2 (sRCF)**.** *Suppose h is a collapsing hash function, and π_{sAoK} is a succinct argument of knowledge as required in Theorem 3.1. This sRCF protocol is parameterized by a probability parameter p and takes a function f described by a classical Turing machine as inputs, whose input length is n and output length is m; these information is public before the beginning of the protocol.*

1. *The client samples bit strings* $x_0, x_1 \leftarrow_r \{0,1\}^n$, $x_0^{(outpad)}, x_1^{(outpad)} \leftarrow_r \{0,1\}^n$, $x_0^{(outpad)} \neq x_1^{(outpad)}$ *and prepares and sends*

$$\frac{1}{\sqrt{2}}(|x_0\rangle |x_0^{(outpad)}\rangle + |x_1\rangle |x_1^{(outpad)}\rangle) \tag{13}$$

   to the server.

2. *The server evaluates f in superposition and gets*

$$\frac{1}{\sqrt{2}}(|x_0\rangle |x_0^{(outpad)}\rangle |f(x_0)\rangle + |x_1\rangle |x_1^{(outpad)}\rangle |f(x_1)\rangle) \tag{14}$$

   *Label these three registers as **inp**, **outpad**, **out** correspondingly.*

   *Then the server does bitwise Hadamard measurement on the **inp** register and gets measurement outcome* $d^{(inp)} \in \{0,1\}^n$ *and the remaining state*

$$\frac{1}{\sqrt{2}}((-1)^{d^{(inp)}\cdot x_0} |x_0^{(outpad)}\rangle |f(x_0)\rangle + (-1)^{d^{(inp)}\cdot x_1} |x_1^{(outpad)}\rangle |f(x_1)\rangle) \tag{15}$$

   The server sends back $d^{(inp)}$ *to the client. The client rejects if* $d^{(inp)}$ *is all-zero.*

3. *The client samples roundtype* $= \mathsf{test}$ *with probability p and sets roundtype* $= \mathsf{comp}$ *otherwise. The two branches of the protocol are designed as follows:*

   - *roundtype* $= \mathsf{test}$*:*

     (a) *The client asks the server measures the **outpad** register on the Hadamard basis; the server performs the operation and sends back the outcome* $d^{(outpad)} \in \{0,1\}^n$.

(b) *The server measures all the* ***out*** *registers on the Hadamard basis and gets* $d^{(out)} \in \{0,1\}^m$.

Note in the honest setting $d^{(out)}$ *satisfies that the following relation evaluates to* 0*:*

$$d^{(inp)}\cdot(x_0+x_1)+d^{(outpad)}\cdot(x_0^{(outpad)}+x_1^{(outpad)})+d^{(out)}\cdot(f(x_0)+f(x_1)) \mod 2 \quad (16)$$

The server computes $c = h(d^{(out)})$ *and sends back* c *to the client.*

(c) *The client sends* $(x_0^{(outpad)}, x_1^{(outpad)})$ *to the server.*

(d) *The client sends* (x_0, x_1) *to the server.*

(e) *Consider the following NP relation:*

$$\big(\underbrace{x_0, x_1 \in \{0,1\}^n, x_0^{(outpad)}, x_1^{(outpad)}, d^{(inp)} \in \{0,1\}^n, d^{(outpad)} \in \{0,1\}^n, c \in range(h)}_{public\ information}, \underbrace{d^{(out)}}_{witness} \in \{0,1\}^m\big) :$$

$$((16) = 0) \wedge (h(d^{(out)}) = c)$$

The client and the server run π_{sAoK} *for this relation, with security parameter being* n*. The honest server could use* $d^{(out)}$ *that it gets previously as the witness to pass the protocol.*

- *If roundtype* = comp*:*
  The client asks the server to measure all the registers in (15) *on the computational basis; the server performs the operations and sends back the outcome* r *of the* ***outpad*** *register. The client checks* $r \in \{x_0^{(outpad)}, x_1^{(outpad)}\}$*. If* $r = x_0^{(outpad)}$*, the client stores* x_0 *and disgards* x_1*. If* $r = x_1^{(outpad)}$*, the client stores* x_1 *and disgards* x_0*.*

Theorem 3.2. *Protocol 2 is complete with completeness error* $\sqrt{p} + \mathsf{negl}(n)$*.*

Proof. The completeness error comes from the test space. Notice that the probability that $d^{(inp)} = 0^n$ is negligibly small, and the completeness of the other steps trivially follow from the protocol description. □

The efficiency is also trivial from the protocol description:

Theorem 3.3. *The communication in Protocol 2 is succinct and all the parties run in polynomial time.*

We formulate the security in the subsection below.

3.3 Security Statement

We will formulate the security for a function family called *inner-product-uniform* function family defined below.

Definition 3.10. Consider a function family $f = (f_n)_{n\in\mathbb{N}}, f_n : \{0,1\}^n \to \{0,1\}^m, m = m(n)$. We say f is inner-product-uniform if for any $(d^{(inp)}, d^{(out)}) \in \{0,1\}^n \times \{0,1\}^m$, $(d^{(inp)}, d^{(out)})$ is not all-zero, there is

$$\Pr_{x\leftarrow_r\{0,1\}^n}[d^{(inp)} \cdot x + d^{(out)} \cdot f(x) \equiv 0 \mod 2] \approx_{\mathsf{negl}(n)} \frac{1}{2}$$

Denote the set of such function family as $\mathcal{F}_{ipu}$.

Theorem 3.4. *For any constant $\delta \in (0,1)$ smaller than a universal constant, constant $p \in (0,1)$ smaller than a universal constant, Protocol 2 with parameter p is sound with soundness $1 - p\delta$ and approximation error $\epsilon = 2\sqrt{p} + \mathsf{poly}(\delta)$ for functions in $\mathcal{F}_{ipu}$.*

A concrete calculation shows $\mathsf{poly}(\delta) = 110\delta^{1/16}$.

Then by the RSPV cut-and-choose amplification in [30], we have:

Corollary 3.5. *For any constant $\epsilon \in (0,1)$, there exists an sRCF under Definition 3.2, 3.3, 3.4, 3.6 for functions in $\mathcal{F}_{ipu}$ with approximation error ϵ.*

4 Security Proof

In this section we prove Theorem 3.4.

4.1 Adversary Modeling and Proof Outline

We begin by introducing several notations for modeling and charactering the adversary's and client's operations.

Notation 4.1. In this work we use purified joint state to track the state appeared during the protocol. That is, we consider the joint state of the client and the server and purify all the states appeared in the protocol by the environment. This allows us to denote states appeared in a protocol as a state in $\mathcal{C} \otimes \mathcal{M} \otimes \mathcal{S} \otimes \mathcal{R}$ (client, message, server, reference (environment)). Furthermore, suppose the input state is $|\varphi\rangle$ then the output state of executing a protocol π starting with this state could be denoted as $\pi \circ |\varphi\rangle$.

Notation 4.2. The joint state that the client and the server create at the first step is denoted as

$$|\varphi^1\rangle = \sum_{x_0, x_1 \in \{0,1\}^n} \sum_{x_0^{(outpad)}, x_1^{(outpad)} \in \{0,1\}^n, x_0^{(outpad)} \neq x_1^{(outpad)}} \frac{1}{2^n \sqrt{2^n(2^n-1)}} \underbrace{|x_0, x_1, x_0^{(outpad)}, x_1^{(outpad)}\rangle}_{\text{client}}$$

$$\otimes \frac{1}{\sqrt{2}} \underbrace{(|x_0\rangle |x_0^{(outpad)}\rangle + |x_1\rangle |x_1^{(outpad)}\rangle)}_{\text{server}}$$

And define its branches $|\varphi_0^1\rangle, |\varphi_1^1\rangle$ as

$$\forall b \in \{0,1\}, |\varphi_b^1\rangle =$$

$$\sum_{x_0, x_1 \in \{0,1\}^n} \sum_{x_0^{(outpad)}, x_1^{(outpad)} \in \{0,1\}^n, x_0^{(outpad)} \neq x_1^{(outpad)}} \frac{1}{2^n \sqrt{2^n(2^n-1)}} |x_0, x_1, x_0^{(outpad)}, x_1^{(outpad)}\rangle$$

$$\otimes \frac{1}{\sqrt{2}} |x_b\rangle |x_b^{(outpad)}\rangle$$

thus $|\varphi_0^1\rangle + |\varphi_1^1\rangle = |\varphi^1\rangle$

Notation 4.3. Suppose the adversary runs in polynomial time, and is described as Adv. The input function f is described by a Turing machine and is in $\mathcal{F}_{ipu}$ (Definition 3.10).

We introduce the following settings and notations, by the time order of the protocol execution.

1. We denote the adversary's operation in the step 2 of Protocol 2 (that is, before returning $d^{(inp)}$ and after getting $|\varphi^1\rangle$) as Adv^2.

2. The operation that the server sends back the value of $\boldsymbol{d}^{(inp)}$ is denoted as $\mathsf{Recv}(\boldsymbol{d}^{(inp)})$.

3. In the case of roundtype $=$ test, the adversary's operation in the 3.a step and 3.b step is denoted as $\mathsf{Adv}^{\mathsf{test}.a}$.

4. The operation that the server sends back the values of $(\boldsymbol{d}^{(outpad)}, \boldsymbol{c})$ is denoted as $\mathsf{Recv}(\boldsymbol{d}^{(outpad)}, \boldsymbol{c})$.

5. The client's sending operation in step 3.c is denoted as $\mathsf{Send}(\boldsymbol{x}_0^{(outpad)}, \boldsymbol{x}_1^{(outpad)})$.

6. The client's sending operation in step 3.d is denoted as $\mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1)$. Suppose the transcript registers that receive $\mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1)$ are denoted as $\boldsymbol{x}_0', \boldsymbol{x}_1'$.

7. The adversary in step e is denoted as Adv^{sAoK}.

Then define the following states that appear during the protocol: Define $|\varphi^2\rangle$ as the output state after the completion of step 2:

$$|\varphi^2\rangle := \mathsf{Recv}(\boldsymbol{d}^{(inp)}) \circ \mathsf{Adv}^2 \circ |\varphi^1\rangle$$

And define $|\varphi_b^2\rangle$ as the output state when the input state is $|\varphi_b^1\rangle$:

$$\forall b \in \{0,1\} : |\varphi_b^2\rangle := \mathsf{Recv}(\boldsymbol{d}^{(inp)}) \circ \mathsf{Adv}^2 \circ |\varphi_b^1\rangle$$

And define the state after the completion of step c conditioned on roundtype $=$ test:

$$|\varphi^{test.c}\rangle := \mathsf{Send}(\boldsymbol{x}_0^{(outpad)}, \boldsymbol{x}_1^{(outpad)}) \circ \mathsf{Recv}(\boldsymbol{d}^{(outpad)}, \boldsymbol{c}) \circ \mathsf{Adv}^{test.a} \circ \mathsf{Recv}(\boldsymbol{d}^{(inp)}) \circ \mathsf{Adv}^2 \circ |\varphi^1\rangle \quad (17)$$

and define $|\varphi_b^{test.c}\rangle$ similarly:

$$|\varphi_b^{test.c}\rangle := \mathsf{Send}(\boldsymbol{x}_0^{(outpad)}, \boldsymbol{x}_1^{(outpad)}) \circ \mathsf{Recv}(\boldsymbol{d}^{(outpad)}, \boldsymbol{c}) \circ \mathsf{Adv}^{test.a} \circ \mathsf{Recv}(\boldsymbol{d}^{(inp)}) \circ \mathsf{Adv}^2 \circ |\varphi_b^1\rangle \quad (18)$$

Notation 4.4. We introduce the following notations for describing the passing conditions. (See Notation 2.1 for notations used below.)

The operator Π_0^{Hada} operates on registers $(\boldsymbol{d}^{(inp)}, \boldsymbol{d}^{(outpad)}, \boldsymbol{d}^{(out)})$ together with $(\boldsymbol{x}_0, \boldsymbol{x}_1)$; it is defined as the projection onto the space that (16) evaluates to 0. Similarly Π_1^{Hada} is defined to be the projection to the subspace that (16) evaluates to 1.

Define $\Pi_{\mathsf{pass}}^{\text{nz}}$ as the projection onto the space that the value of $\boldsymbol{d}^{(inp)}$ is not all-zero. Define $\Pi_{\mathsf{fail}}^{\text{nz}}$ as the projection onto the space that $\boldsymbol{d}^{(inp)}$ is all-zero.

The operator $\Pi_{\mathsf{pass}}^{\text{hash}}$ operates on registers $\boldsymbol{d}^{(out)}$ and $\boldsymbol{c}$; it is defined to be the projection onto the space that $h(d^{(out)}) = c$.

With these notations, π_{sAoK} is to verify that $d^{(out)}$ falls within $\Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{hash}}$.

As an example, the passing condition in Toy Protocol 1 could be denoted as $\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_0^{\text{Hada}}$.

Notation 4.5. Use π_{test} to denote the execution of Protocol 2 where roundtype $=$ test is chosen with probability 1.

Below we give a step-by-step overview of our proof. For a top-down overview, we refer to Section 1.2.3.

1. In Section 4.2 we analyze the π_{sAoK} protocol.
2. In Section 4.3 we prove a lemma about amplification of hash function preimage extraction, as discussed in Section 1.2.2.
3. In Section 4.4 we show starting from the state $\mathsf{Send}(\boldsymbol{x}_b)\circ|\varphi_b^{test.c}\rangle$ (that is, the server does not know key x_{1-b}), a hash preimage of c could be efficiently extracted on the server-side with probability close to $|\,|\varphi_b^{test.c}\rangle\,|^2$.
4. In Section 4.5 we show starting from the state $|\varphi^{test.c}\rangle$ (that is, the server does not know any key), a hash preimage of c could be efficiently extracted on the server-side with probability close to 1.
5. In Section 4.6 we construct the simulator and complete the proof.

4.2 Analysis of π_{sAoK}

Lemma 4.1. *In protocol π_{test} suppose a polynomial time adversary Adv makes the client accept with probability $\geq 1-\delta$. Expanding the π_{test} execution after step 3.c:*

$$|\Pi_{\mathsf{pass}}\circ\pi_{sAoK}^{\mathsf{Adv}^{sAoK}}\circ\mathsf{Send}(\boldsymbol{x}_0,\boldsymbol{x}_1)\circ|\varphi^{test.c}\rangle\,|^2\geq 1-\delta \tag{19}$$

then there exists a polynomial time computable server-side unitary E such that

$$|\Pi_0^{Hada}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash}\circ E\circ\mathsf{Send}(\boldsymbol{x}_0,\boldsymbol{x}_1)\circ|\varphi^{test.c}\rangle\,|^2\geq 1-2\delta-\mathsf{negl}(n) \tag{20}$$

Proof. Apply the knowledge extraction property of π_{sAoK} (Theorem 3.1). Notice the π_{sAoK} is designed to verify whether the server holds $d^{(out)}$ that satisfies the conditions of $\Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}$, translating (12) to the concrete operations here completes the proof. □

4.3 Amplification Lemma of Hash Preimage Extraction

As a preparation we state the following lemma.

Lemma 4.2. *Suppose U is a polynomial time server-side operation, $b\in\{0,1\}$, $|\varphi_b\rangle$ is an efficiently preparable purified joint state on the setting of Notation 4.3 whose server-side state is independent to the value of $\boldsymbol{x}_{1-b}$, there is*

$$|\Pi_0^{Hada}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash}\circ U\circ\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash}|\varphi_b\rangle\,|\leq\frac{1}{\sqrt{2}}|\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash}|\varphi_b\rangle\,|+\mathsf{negl}(n)$$

Proof. Denote COPY as an operation that copies the value of the $\boldsymbol{d}^{(out)}$ register to an auxiliary register $\boldsymbol{aux}$ that is not touched by any operation above. By the collapsing property there is

$$|\Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}\circ U\circ\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}|\varphi\rangle\,|\approx_{\mathsf{negl}(n)}|\Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}\circ U\circ\mathsf{COPY}\circ\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}\circ|\varphi\rangle\,| \tag{21}$$

Then by collision-resistance there is

$$(21)\approx_{\mathsf{negl}(n)}|\Pi_0^{\text{Hada}}\Pi_{\boldsymbol{aux}=\boldsymbol{d}^{(out)}}\circ U\circ\mathsf{COPY}\circ\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}\circ|\varphi\rangle\,| \tag{22}$$

By the inner-product-uniform property of f we know, for each $(d^{(inp)}, d^{(out)}) \in \{0,1\}^n \times \{0,1\}^m$ that is not all-zero, the number of x that falls within the space of Π_0^{Hada} is negligibly close to half of the domain. Formally, we have

$$|\Pi_0^{\text{Hada}}\Pi_{\boldsymbol{aux}=\boldsymbol{d}^{(out)}}\circ U\circ\mathsf{COPY}\circ\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}}\circ|\varphi\rangle\,| \leq \frac{1}{\sqrt{2}}|\Pi_{\boldsymbol{aux}=\boldsymbol{d}^{(out)}}\circ U\circ\mathsf{COPY}\circ\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}}\circ|\varphi\rangle\,|+\mathsf{negl}(n) \tag{23}$$

This completes the proof. □

The following lemma is the main lemma of this subsection.

Lemma 4.3. *Consider the setting of Notation 4.3. Suppose* Adv *is a polynomial time adversary,* $b \in \{0,1\}$*. Consider* $|\varphi_b^{test.c}\rangle$ *as defined in* (18)*. Suppose there exists a server-side efficient unitary* U *such that*

$$|\Pi_0^{Hada}\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}}\circ U\circ\mathsf{Send}(\boldsymbol{x}_0,\boldsymbol{x}_1)\circ|\varphi_b^{test.c}\rangle\,|^2\geq q \tag{24}$$

where q *is a constant. Then for any constant* $q' \in (q, 2q)$*, there exists an efficiently computable server-side unitary* V_b *such that*

$$|\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}}\circ V_b\circ\mathsf{Send}(\boldsymbol{x}_b)\circ|\varphi_b^{test.c}\rangle\,|^2\geq q'-\mathsf{negl}(n) \tag{25}$$

We first introduce a notation that will be useful.

Notation 4.6. Suppose an operation U operates on the server-side, which has classical (read-only) access to the transcript registers including $\boldsymbol{x}_0', \boldsymbol{x}_1'$. Use $U[\boldsymbol{u},\boldsymbol{v}]$ to denote the following operation. Suppose $\boldsymbol{u}$, $\boldsymbol{v}$ are initially prepared to hold uniform superpositions of the same size of $\boldsymbol{x}_0, \boldsymbol{x}_1$. Then $U[\boldsymbol{u},\boldsymbol{v}]$ denotes $\mathsf{SWAP}((\boldsymbol{x}_0',\boldsymbol{x}_1'),(\boldsymbol{u},\boldsymbol{v}))\circ U\circ\mathsf{SWAP}((\boldsymbol{x}_0',\boldsymbol{x}_1'),(\boldsymbol{u},\boldsymbol{v}))$, which means, replacing all the usage of $(\boldsymbol{x}_0,\boldsymbol{x}_1)$ in U by the values of $(\boldsymbol{u},\boldsymbol{v})$.

Proof. Without loss of generality consider $b = 0$.

Initialize server-side registers $\boldsymbol{t}^{(1)}$, $\boldsymbol{t}^{(2)}$, $\cdots$, $\boldsymbol{t}^{(t)}$ where $t = 10/(2q-q')^2$. Each register holds a superposition that is of the same size as $\boldsymbol{x}_1$. Note that $|\varphi_0^{test.c}\rangle$ does not depend on $\boldsymbol{x}_1$, then (24) implies, for each i,

$$|\Pi_0^{\text{Hada}}\Pi^{\text{hash}}_{\mathsf{pass}}\circ U[\boldsymbol{x}_0',\boldsymbol{t}^{(i)}]\circ\mathsf{Send}(\boldsymbol{x}_0,\boldsymbol{t}^{(1)}\cdots\boldsymbol{t}^{(t)})\circ\Pi^{\text{nz}}_{\mathsf{pass}}|\varphi_0^{test.c}\rangle\,|^2\geq q \tag{26}$$

(where we abuse the notation a little bit to use $\mathsf{Send}(\boldsymbol{t}^{(1)}\cdots\boldsymbol{t}^{(t)})$ to denote the operation that initializes the $\boldsymbol{t}$ register as replacements of $\boldsymbol{x}_1$.)

Now we construct V_0 that satisfies (25).

Initialize $\boldsymbol{r}^{(1)}$, $\boldsymbol{r}^{(2)}$, $\cdots$, $\boldsymbol{r}^{(t)}$ where each of them is a 1-bit register initialized to 0.

Define V_0 as follows:

1. For $i = 1, 2\cdots t$:
   (a) If $i = 1$ apply $U[\boldsymbol{x}_0',\boldsymbol{t}^{(1)}]$. Otherwise controlled on $\boldsymbol{r}^{(i-1)}$ is in state 0, apply controlled-$U[\boldsymbol{x}_0',\boldsymbol{t}^{(i)}]$.
   (b) Flip $\boldsymbol{r}^{(i)}$ to 1 if the current state falls in $\Pi^{\text{hash}}_{\mathsf{pass}}$.
   (c) Use $V_0^{(\sim i.a)}$ to denote all the operations that V_0 has done by the end of the i-th round of step a. Controlled on $\boldsymbol{r}^{(i)}$ is in state 0, apply controlled-$(V^{(\sim i.a)})^{-1}$.

For further proofs and intuitive understanding, define $|\phi^{(i)}\rangle, |\chi^{(i)}\rangle$ as follows:

$$|\phi^{(i)}\rangle = \Pi^{\text{hash}}_{\mathsf{pass}} \circ V_0^{(\sim i.a)} \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{t}^{(1)} \cdots \boldsymbol{t}^{(t)}) \circ \Pi^{\text{nz}}_{\mathsf{pass}} |\varphi_0^{test.c}\rangle,$$

$$|\chi^{(i)}\rangle = (\mathsf{I} - \Pi^{\text{hash}}_{\mathsf{pass}}) \circ V_0^{(\sim i.a)} \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{t}^{(1)} \cdots \boldsymbol{t}^{(t)}) \circ \Pi^{\text{nz}}_{\mathsf{pass}} |\varphi_0^{test.c}\rangle$$

Then the construction of V_0 intuitively reads as follows. For a round count number i, note that without step (b), all the operations that V_0 has done by this time will be rewinded by step i.c and the state will be mapped back to the initial state $|\varphi_0^{test.c}\rangle$. Now, with step b, the hash preimages that V_0 have extracted so far are preserved and will not be affected by the later operations, and what's "rewinded back" is just the $|\chi\rangle$ part. So why do we want to take it back? In the next round we will initialize a new $\boldsymbol{t}^{(i)}$ (note i has been increased) and running U could possibly produce more component d that falls within $\Pi^{\text{hash}}_{\mathsf{pass}}$. Repeating this for t times and we claim this is sufficient for extracting preimage in $\Pi^{\text{hash}}_{\mathsf{pass}}$ with probability q'.

Below we prove V_0 indeed achieves what we want.

Denote $a_i := |\,|\phi^{(i)}\rangle\,|$. Then there is:

- $a_i \leq a_{i+1}$;
- If $a_i < q'$, $a_{i+1} > a_i + \frac{1}{10}(2q - q')^2 - \mathsf{negl}(n)$

The first statement above is trivial. To prove the second statement, we make use of Lemma 4.3 above. Once we have both statement, by elementary calculation $a_t \geq q' - \mathsf{negl}(n)$ which completes the proof.

(26) says

$$|\Pi_0^{\text{Hada}} \Pi^{\text{nz}}_{\mathsf{pass}} \Pi^{\text{hash}}_{\mathsf{pass}} \circ U[\boldsymbol{x}'_0, \boldsymbol{t}^{(i+1)}] \circ (V_0^{(\sim i.a)})^{-1} \circ (|\phi^{(i)}\rangle + |\chi^{(i)}\rangle)|^2 \geq q \tag{27}$$

By Lemma 4.3 there is

$$|\Pi_0^{\text{Hada}} \Pi^{\text{nz}}_{\mathsf{pass}} \Pi^{\text{hash}}_{\mathsf{pass}} \circ U[\boldsymbol{x}'_0, \boldsymbol{t}^{(i+1)}] \circ (V_0^{(\sim i.a)})^{-1} \circ |\phi^{(i)}\rangle\,|^2 \leq \frac{1}{2}|\,|\phi^{(i)}\rangle\,| + \mathsf{negl}(n) < \frac{1}{2}q' + \mathsf{negl}(n) \tag{28}$$

Combining them we get

$$|\Pi_0^{\text{Hada}} \Pi^{\text{nz}}_{\mathsf{pass}} \Pi^{\text{hash}}_{\mathsf{pass}} \circ U[\boldsymbol{x}'_0, \boldsymbol{t}^{(i+1)}] \circ (V_0^{(\sim i.a)})^{-1} \circ |\chi^{(i)}\rangle\,|^2 \geq (\sqrt{q} - \sqrt{\frac{1}{2}q'})^2 - \mathsf{negl}(n) \geq \frac{1}{10}(2q-q')^2 - \mathsf{negl}(n) \tag{29}$$

This means in the next round the square norm of the state projected on $\Pi^{\text{hash}}_{\mathsf{pass}}$ increases by this value, which completes the proof. □

4.4 Analyzing a Single Branch

Lemma 4.4. *Consider the setting of Notation 4.3. Suppose a polynomial time adversary* Adv *makes* (19) *hold. Then for each* $b \in \{0, 1\}$*, there exists an efficiently computable server-side unitary* V_b *such that*

$$|\Pi^{nz}_{\mathsf{pass}} \Pi^{hash}_{\mathsf{pass}} \circ V_b \circ \mathsf{Send}(\boldsymbol{x}_b) \circ |\varphi_b^{test.c}\rangle\,|^2 \geq \frac{1}{2} - 40\delta - \mathsf{negl}(n)$$

That is, V_b *does not have access to* $\boldsymbol{x}_{1-b}$.

Proof. Consider E as given in Lemma 4.1.

First by Lemma 4.3, notice the left hand side of (25) is no more than $|\,|\varphi_b^{test.c}\rangle\,|^2 = \frac{1}{2}$, we have

$$|\Pi_0^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}}\Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ E \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1) \circ |\varphi_b^{test.c}\rangle\,|^2 \leq \frac{1}{4} + \delta + \mathsf{negl}(n) \tag{30}$$

Combining (30) for $b = 0, b = 1$ and (20) we have, for each $b = 0$, $b = 1$,

$$|\Pi_0^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}}\Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ E \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1) \circ |\varphi_b^{test.c}\rangle\,|^2 \geq \frac{1}{4} - 10\delta - \mathsf{negl}(n) \tag{31}$$

Applying Lemma 4.3 again completes the proof. □

4.5 Analyzing Both Branches

Lemma 4.5. *Consider the setting of Notation 4.3. Suppose a polynomial time adversary* Adv *makes* (19) *hold. Then there exists a polynomial time operation W such that*

$$|\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash} \circ W \circ |\varphi^{test.c}\rangle\,|^2 \geq 1 - 400\delta^{1/8} - \mathsf{negl}(n) \tag{32}$$

Furthermore,

$$\Pi_0^{Hada}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash} \circ W \circ |\varphi_0^{test.c}\rangle \approx_{10\delta^{1/16}} \Pi_0^{Hada}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash} \circ W \circ |\varphi_1^{test.c}\rangle \tag{33}$$

$$\Pi_1^{Hada}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash} \circ W \circ |\varphi_0^{test.c}\rangle \approx_{6\delta^{1/8}} -\Pi_1^{Hada}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash} \circ W \circ |\varphi_1^{test.c}\rangle \tag{34}$$

$$\forall b \in \{0,1\}, |(\mathsf{I} - \Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{hash}) \circ W \circ |\varphi_b^{test.c}\rangle\,| \leq 20\delta^{1/16} + \mathsf{negl}(n) \tag{35}$$

Proof. We construct W in two steps. We first prove an operation V_0 has such property with access to $\boldsymbol{x}_0$, then construct W that does not need any $\boldsymbol{x}$.

Step 1 By Lemma 4.4 there exists a polynomial time operation V_0 such that

$$|\Pi_{\mathsf{pass}}^{\mathrm{nz}}\Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle\,|^2 \geq \frac{1}{2} - 40\delta - \mathsf{negl}(n) \tag{36}$$

We would like to prove

$$|\Pi_{\mathsf{pass}}^{\mathrm{nz}}\Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle\,|^2 > \frac{1}{2} - \mathsf{poly}(\delta) - \mathsf{negl}(n) \tag{37}$$

We prove it by analyzing different components of (37) step by step.

(Analyzing the $\Pi_1^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}}$ component) (20) in Lemma 4.1 implies

$$\Pi_0^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}}\Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ E \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1) \circ |\varphi^{test.c}\rangle \approx_{\sqrt{2\delta}} \Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ E \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1) \circ |\varphi^{test.c}\rangle \tag{38}$$

By the collapsing property of hash functions, it's hard to transform component in $\Pi_0^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}}\Pi_{\mathsf{pass}}^{\mathrm{hash}}$ to $(\mathsf{I} - \Pi_0^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}})\Pi_{\mathsf{pass}}^{\mathrm{hash}}$. (Here the transformation being considered is $V_0 \circ E^{-1}$.) Formally speaking, (38) and the collapsing property implies

$$|(\mathsf{I} - \Pi_0^{\mathrm{Hada}}\Pi_{\mathsf{pass}}^{\mathrm{nz}})\Pi_{\mathsf{pass}}^{\mathrm{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0, \boldsymbol{x}_1) \circ |\varphi^{test.c}\rangle\,|^2 \leq 3\sqrt{\delta} + \mathsf{negl}(n) \tag{39}$$

Since V_0 does not touch $\boldsymbol{x}_1, \boldsymbol{x}'_1$, we have

$$|(\mathsf{I} - \Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}})\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi^{test.c}\rangle\,|^2 \leq 3\sqrt{\delta} + \mathsf{negl}(n) \tag{40}$$

(40) could be re-written as

$$(\mathsf{I} - \Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}})\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle \approx_{3\sqrt{\delta}} -(\mathsf{I} - \Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}})\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle \tag{41}$$

which implies the cancellation of the $\Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}$ component of the two branches as a collorary:

$$\Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle \approx_{3\sqrt{\delta}} -\Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle \tag{42}$$

(Analyzing the $\Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}$ component) Starting from (36), by Lemma 4.2 we know

$$|\Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle\,|^2 \geq \frac{1}{4} - 20\delta - \mathsf{negl}(n) \tag{43}$$

Then together with (42) we know

$$|\Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle\,|^2 \geq \frac{1}{4} - 6\sqrt{\delta} - 30\delta - \mathsf{negl}(n) \tag{44}$$

(43)(44) imply

$$|(\mathsf{I} - \Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}) \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle\,|^2 \leq \frac{1}{4} + 20\delta + \mathsf{negl}(n) \tag{45}$$

$$|(\mathsf{I} - \Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}) \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle\,|^2 \leq \frac{1}{4} + 6\sqrt{\delta} + 30\delta + \mathsf{negl}(n) \tag{46}$$

Summing up the four states appeared in (45)(46)(42) should result in a state of norm 1. Thus combining (45)(46)(42) we get

$$(\mathsf{I} - \Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}) \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle \approx_{6\delta^{1/4}} (\mathsf{I} - \Pi_1^{\text{Hada}}\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}}) \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle \tag{47}$$

which implies

$$\Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle \approx_{6\delta^{1/4}} \Pi_0^{\text{Hada}}\Pi_{\mathsf{pass}}^{nz}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle \tag{48}$$

(Analyzing the Π_{fail}^{nz} component) From (36) we get

$$|\Pi_{\mathsf{fail}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_0^{test.c}\rangle\,| \leq \sqrt{40\delta} + \mathsf{negl}(n) \tag{49}$$

Combining it with (47) we get

$$|\Pi_{\mathsf{fail}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle\,| \leq 6\delta^{1/4} + \sqrt{40\delta} + \mathsf{negl}(n) \tag{50}$$

(Completing the proof of (37)) Combining (36)(42)(48)(50) we get

$$|\Pi_{\mathsf{pass}}^{\text{nz}}\Pi_{\mathsf{pass}}^{\text{hash}} \circ V_0 \circ \mathsf{Send}(\boldsymbol{x}_0) \circ |\varphi_1^{test.c}\rangle\,|^2 \geq \frac{1}{2} - 48\delta^{1/4} - \mathsf{negl}(n) \tag{51}$$

Step 2 Now consider W defined as follows: Initialize $\boldsymbol{t}$ register that is of the same size as the $\boldsymbol{x}_0'$ register, run $V_0[\boldsymbol{t}]$.

Since in (51) the server-side of the $\boldsymbol{x}_1$ branch does not depend on $\boldsymbol{x}_0$, (51) implies

$$|\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_1^{test.c}\rangle\,|^2 \geq \frac{1}{2} - 48\delta^{1/4} - \mathsf{negl}(n) \tag{52}$$

We would like prove

$$|\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_0^{test.c}\rangle\,|^2 \geq \frac{1}{2} - \mathsf{poly}(\delta) - \mathsf{negl}(n)$$

We could repeat the argument above for the leap from branch $\boldsymbol{x}_1$ to $\boldsymbol{x}_0$, and (52) plays the role of (36). Similarly we could prove $|(\mathsf{I} - \Pi^{\text{Hada}}_0\Pi^{\text{nz}}_{\mathsf{pass}})\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi^{test.c}\rangle\,|$ is small, and prove $\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_1^{test.c}\rangle$ distributes with almost equal norms on subspace Π^{Hada}_0 and Π^{Hada}_1. Then analogous to (42)(48) we have

$$\Pi^{\text{Hada}}_0\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_0^{test.c}\rangle \approx_{10\delta^{1/16}} \Pi^{\text{Hada}}_0\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_1^{test.c}\rangle$$

$$\Pi^{\text{Hada}}_1\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_0^{test.c}\rangle \approx_{6\delta^{1/8}} -\Pi^{\text{Hada}}_1\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi_1^{test.c}\rangle$$

these two together with (52) imply

$$\forall b \in \{0,1\}, |(\mathsf{I} - \Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}}) \circ W \circ |\varphi_b^{test.c}\rangle\,| \leq 20\delta^{1/16} + \mathsf{negl}(n)$$

which together imply

$$|\Pi^{\text{nz}}_{\mathsf{pass}}\Pi^{\text{hash}}_{\mathsf{pass}} \circ W \circ |\varphi^{test.c}\rangle\,|^2 \geq 1 - 400\delta^{1/8} - \mathsf{negl}(n)$$

□

A corollary discusses extracting $d^{(out)}$ from $|\varphi^2\rangle$, which is easier to use in later proof:

Corollary 4.6. *Consider the setting of Notation 4.3. Suppose a polynomial time adversary* Adv *makes* (19) *hold. Then there exists a polynomial time operation* W' *that takes the output states of step 2 of the protocol as inputs, operates on the server-side with classical (read-only) access to the* $\boldsymbol{x}_0^{(outpad)}, \boldsymbol{x}_1^{(outpad)}$ *registers, and writes output states on* $\boldsymbol{d}^{(outpad)}, \boldsymbol{d}^{(out)}$ *registers such that*

$$|\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}} \circ W' \circ |\varphi^2\rangle\,|^2 \geq 1 - 400\delta^{1/8} - \mathsf{negl}(n) \tag{53}$$

Furthermore,

$$\Pi^{Hada}_0\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}} \circ W' \circ |\varphi_0^2\rangle \approx_{10\delta^{1/16}} \Pi^{Hada}_0\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}} \circ W' \circ |\varphi_1^2\rangle \tag{54}$$

$$\Pi^{Hada}_1\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}} \circ W' \circ |\varphi_0^2\rangle \approx_{6\delta^{1/8}} -\Pi^{Hada}_1\Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}} \circ W' \circ |\varphi_1^2\rangle \tag{55}$$

$$\forall b \in \{0,1\}, |(\mathsf{I} - \Pi^{nz}_{\mathsf{pass}}\Pi^{hash}_{\mathsf{pass}}) \circ W' \circ |\varphi_b^2\rangle\,| \leq 20\delta^{1/16} + \mathsf{negl}(n) \tag{56}$$

4.6 Constructing the Simulator

In this subsection we construct the simulator and prove Theorem 3.4. The simulator will simulate the comp type of round; for analyzing this roundtype we introduce the following notation.

Notation 4.7. For an adversary Adv of Protocol 2, denote the adversarial operation in the comp roundtype as $\mathsf{Adv}^{\mathsf{comp}}$. Recall that in Protocol 2 the operation that follows $\mathsf{Adv}^{\mathsf{comp}}$ is $\mathsf{Recv}(\boldsymbol{r})$; suppose $\mathsf{Adv}^{\mathsf{comp}}$ writes its output in its own register $\boldsymbol{r}'$ and $\mathsf{Recv}(\boldsymbol{r})$ is implemented as copying the values of register $\boldsymbol{r}'$ to a transcript register denoted as $\boldsymbol{r}$.

As a preparation, we state and prove the following lemma.

Lemma 4.7. *For any polynomial time adversary* Adv *in Protocol 2,*

$$\forall b \in \{0,1\}, \Pi_{\boldsymbol{r}=\boldsymbol{x}_b^{(outpad)}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2\rangle \approx_{\mathsf{negl}(n)} \Pi_{\boldsymbol{r}=\boldsymbol{x}_b^{(outpad)}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2_b\rangle$$

Recall $|\varphi^2\rangle$ is defined in Notation 4.3.

Proof. This is reduced to proving

$$\Pi_{\boldsymbol{r}=\boldsymbol{x}_b^{(outpad)}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2_{1-b}\rangle \approx_{\mathsf{negl}(n)} 0$$

Since the server-side information of $|\varphi^2_{1-b}\rangle$ is independent to the values of $\boldsymbol{x}_b^{(outpad)}$, projecting to the right value has probability only $\frac{1}{2^n-1}$. This completes the proof. □

Proof of Theorem 3.4. Below we analyze the protocol, construct the simulator and prove Theorem 3.4. As before we use the notations in Notations 4.3.

Analyzing protocol Suppose the protocol is executed against an efficient adversary Adv and passes with probability $\geq 1 - p\delta$. This implies

$$\mathrm{tr}(\Pi_{\mathsf{pass}}(\pi^{\mathsf{Adv}}_{test}(1^n))) > 1 - \delta$$

By Corollary 4.6 there exists a polynomial time server-side operation W' such that (53)-(56) hold.

Constructing Sim The input state of Sim is

$$\sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} \underbrace{|x\rangle}_{\text{client}} \otimes \underbrace{|f(x)\rangle}_{\text{server}} \tag{57}$$

and Sim only operates on the server-side (including writing into the transcript registers).

Our construction is as follows:

1. Instantiate $\tilde{\boldsymbol{x}}_0^{(outpad)}, \tilde{\boldsymbol{x}}_1^{(outpad)}$ registers whose values are sampled uniformly randomly in $\{0,1\}^n \times \{0,1\}^n$ such that $\tilde{x}_0^{(outpad)} \neq \tilde{x}_1^{(outpad)}$.

   Sample $b \leftarrow_r \{0,1\}$ with equal probability.

   - If $b = 0$:

     (a) Write $\tilde{x}_0^{(outpad)}$ to the register $\boldsymbol{r}$.

(b) Instantiate $\tilde{\boldsymbol{x}}_1$ with uniform superpositions in $\{0,1\}^n$.
Prepare the state

$$\underbrace{|\tilde{x}_1\rangle}_{\boldsymbol{inp}}\underbrace{|\tilde{x}_1^{(outpad)}\rangle}_{\boldsymbol{outpad}}\underbrace{|f(\tilde{x}_1)\rangle}_{\boldsymbol{out}} \tag{58}$$

where $\tilde{x}_1$ is the value of register $\tilde{\boldsymbol{x}}_1$, $\tilde{x}_1^{(outpad)}$ is the value of register $\tilde{\boldsymbol{x}}_1^{(outpad)}$.

(c) Apply operator Adv^2.

(d) Apply W'.

(e) As described in Collorary 4.6 the registers $\boldsymbol{d}^{(outpad)}, \boldsymbol{d}^{(out)}$ have been initialized and written in. Add a (-1) phase on the space that satisfies

$$d^{(outpad)} \cdot (\tilde{x}_0^{(outpad)} + \tilde{x}_1^{(outpad)}) + d^{(out)} \cdot (f(x) + f(\tilde{x}_1)) \equiv 1 \mod 2 \tag{59}$$

This could be done since the Sim has access to all the information appeared in (59). Denote the operation in this step as $\mathsf{FlipSign}_{(59)}$.

(f) Apply $(W')^{-1}$.

(g) Apply $\mathsf{Adv}^{\mathsf{comp}}$. This initializes the $\boldsymbol{r}'$ register as described in Notation 4.7.

(h) Write pass to the flag register if $\boldsymbol{r}' = \boldsymbol{r}$.

- If $b = 1$: (Note it's the same as $b = 0$ case except changing the subscript bit.)

(a) Write $\tilde{x}_1^{(outpad)}$ to the register $\boldsymbol{r}$.

(b) Instantiate $\tilde{\boldsymbol{x}}_0$ with uniform superpositions in $\{0,1\}^n$.
Prepare the state

$$|\tilde{x}_0\rangle\,|\tilde{x}_0^{(outpad)}\rangle\,|f(\tilde{x}_0)\rangle \tag{60}$$

where $\tilde{x}_0$ is the value of register $\tilde{\boldsymbol{x}}_0$, $\tilde{x}_0^{(outpad)}$ is the value of register $\tilde{\boldsymbol{x}}_0^{(outpad)}$.

(c) Apply operator Adv^2.

(d) Apply W'.

(e) Add a (-1) phase on the space that satisfies

$$d^{(outpad)} \cdot (\tilde{x}_0^{(outpad)} + \tilde{x}_1^{(outpad)}) + d^{(out)} \cdot (f(x) + f(\tilde{x}_0)) \equiv 1 \mod 2 \tag{61}$$

(f) Apply $(W')^{-1}$.

(g) Apply $\mathsf{Adv}^{\mathsf{comp}}$. Recall Notation 4.7 that the temporary output is in the $\boldsymbol{r}'$ register.

(h) Write pass to the flag register if $\boldsymbol{r}' = \boldsymbol{r}$.

2. Disgard registers $\tilde{\boldsymbol{x}}_0^{(outpad)}, \tilde{\boldsymbol{x}}_1^{(outpad)}$, and $\tilde{\boldsymbol{x}}_0$ or $\tilde{\boldsymbol{x}}_1$.

As a notation preparation, the operator in the $b = 0$ branch of the simulator as Sim_0 and denote the $b = 1$ branch as Sim_1. (Thus $\mathsf{Sim} = \mathsf{Sim}_0 + \mathsf{Sim}_1$.)

Analyzing Sim Our goal is to prove

$$\Pi_{\mathsf{pass}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2\rangle \approx^{ind}_{\epsilon'} \Pi_{\mathsf{pass}}(\mathsf{Sim}(\text{equation (57)})) \tag{62}$$

For some $\epsilon' = \mathsf{poly}(\delta)$. Since the distinguisher's access to the $\boldsymbol{r}$ register is read-only, and the passing condition on the left hand side requires that $\boldsymbol{r} \in \{\boldsymbol{x}_0^{(outpad)}, \boldsymbol{x}_1^{(outpad)}\}$ and the passing condition on the right hand side requires $\boldsymbol{r}' = \boldsymbol{r} \in \{\boldsymbol{x}_0^{(outpad)}, \boldsymbol{x}_1^{(outpad)}\}$, (62) is reduced to proving

$$\Pi_{\boldsymbol{r}=\boldsymbol{x}_0^{(outpad)}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2\rangle \approx^{ind}_{\epsilon'/2} \Pi_{\boldsymbol{r}'=\boldsymbol{r}=\tilde{\boldsymbol{x}}_0^{(outpad)}}(\mathsf{Sim}(\text{equation (57)})) \tag{63}$$

$$\Pi_{\boldsymbol{r}=\boldsymbol{x}_1^{(outpad)}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2\rangle \approx^{ind}_{\epsilon'/2} \Pi_{\boldsymbol{r}'=\boldsymbol{r}=\tilde{\boldsymbol{x}}_1^{(outpad)}}(\mathsf{Sim}(\text{equation (57)})) \tag{64}$$

Without loss of generality we only consider (63). By Lemma 4.7 and the construction of Sim this is further reduced to

$$\Pi_{\boldsymbol{r}=\boldsymbol{x}_0^{(outpad)}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi_0^2\rangle \approx^{ind}_{\epsilon'/2} \Pi_{\boldsymbol{r}'=\boldsymbol{r}=\tilde{\boldsymbol{x}}_0^{(outpad)}}(\mathsf{Sim}_0(\text{equation (57)})) \tag{65}$$

Let's open the inner procedure of Sim_0. Sim_0 instantiates a series of registers for simulating the client-side registers appeared in the real protocol. In more detail, as the name suggests,

$$\tilde{\boldsymbol{x}}_0^{(outpad)} \text{ simulates } \boldsymbol{x}_0^{(outpad)},\ \tilde{\boldsymbol{x}}_1^{(outpad)} \text{ simulates } \boldsymbol{x}_1^{(outpad)},\ \tilde{\boldsymbol{x}}_1 \text{ simulates } \boldsymbol{x}_1 \tag{66}$$

Thus the output of step d of the simulator, $W' \circ \mathsf{Adv}^2 \circ (58)$, is equal to $W' \circ |\varphi_1^2\rangle$ up to (66).

Now we move to analyze $\mathsf{FlipSign}$ in step e. Denote the operation that adds a (-1) phase on the space that $d^{(inp)} \cdot (x + \tilde{x}_1) \equiv 1 \mod 2$ as $\mathsf{FlipSign}_{d^{(inp)}\cdot(x+\tilde{x}_1) \mod 2}$. Then by Corollary 4.6 there is

$$W' |\varphi_0^2\rangle \approx_{110\delta^{1/16}} \mathsf{FlipSign}_{d^{(inp)}\cdot(x+\tilde{x}_1) \mod 2} \circ \mathsf{FlipSign}_{(59)} \circ W' \circ \mathsf{Adv}^2 \circ (58) \text{ (up to (66))}$$

Since the distinguisher only has classical access to $d^{(inp)}, x, \tilde{x}_1$, the operation of $\mathsf{FlipSign}_{d^{(inp)}\cdot(x+\tilde{x}_1) \mod 2}$ is undetectable. Thus

$$W' |\varphi_0^2\rangle \approx^{ind}_{110\delta^{1/16}} \mathsf{FlipSign}_{(59)} \circ W' \circ \mathsf{Adv}^2 \circ (58) \text{ (up to (66))}$$

Thus

$$\mathsf{Adv}^{\mathsf{comp}} \circ |\varphi_0^2\rangle \approx^{ind}_{110\delta^{1/16}} \mathsf{Adv}^{\mathsf{comp}} \circ (W')^{-1} \circ \mathsf{FlipSign}_{(59)} \circ W' \circ \mathsf{Adv}^2 \circ (58) \text{ (up to (66))}$$

where the left hand side is the state appeared during the real protocol and the right hand side is almost the operation of Sim_0. Finally we note the passing condition $\boldsymbol{r}' = \boldsymbol{r}$ translates to $\boldsymbol{r}_0 = \boldsymbol{x}_0^{(outpad)}$ which completes the proof of (65). Thus (62) is proved.

Finally note that in Protocol 2 with probability p the comp round is not chosen. (62) only considers the comp roundtype; considering the test part the approximate indistinguishability expression becomes

$$\Pi_{\mathsf{pass}} \circ \mathsf{Recv}(\boldsymbol{r}) \circ \mathsf{Adv}^{\mathsf{comp}} \circ |\varphi^2\rangle \approx^{ind}_{\epsilon} \Pi_{\mathsf{pass}}(\mathsf{Sim}(\text{equation (57)})), \quad \epsilon = 2\sqrt{p} + 110\delta^{1/16} \tag{67}$$

This completes the proof. □

References

[1] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2), may 2012.

[2] James Bartusek. Secure quantum computation with classical communication. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 1–30, Cham, 2021. Springer International Publishing.

[3] James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. One-way functions imply secure computation in a quantum world. *ArXiv*, abs/2011.13486, 2020.

[4] James Bartusek, Yael Tauman Kalai, Alex Lombardi, Fermi Ma, Giulio Malavolta, Vinod Vaikuntanathan, Thomas Vidick, and Lisa Yang. Succinct classical verification of quantum computation. In *IACR Cryptology ePrint Archive*, 2022.

[5] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. 2022. `https://eprint.iacr.org/2022/1178`.

[6] Charles H. Bennett, David P. DiVincenzo, Peter W. Shor, John A. Smolin, Barbara M. Terhal, and William K. Wootters. Remote state preparation. *Phys. Rev. Lett.*, 87:077902, Jul 2001.

[7] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 439–448, New York, NY, USA, 2015. Association for Computing Machinery.

[8] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 320–331, 2018.

[9] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In *IACR Cryptol. ePrint Arch.*, 2020.

[10] A. Chiesa, F. Ma, N. Spooner, and M. Zhandry. Post-quantum succinct arguments: Breaking the quantum rewinding barrier. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 49–58, Los Alamitos, CA, USA, feb 2022. IEEE Computer Society.

[11] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. Qfactory: Classically-instructed remote secret qubits preparation. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 615–645. Springer, 2019.

[12] Alexandru Gheorghiu and Thomas Vidick. Computationally-secure and composable remote state preparation. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1024–1033. IEEE Computer Society, 2019.

[13] Alex B. Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious transfer is in miniqcrypt. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 531–561, Cham, 2021. Springer International Publishing.

[14] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 60–73, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[16] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992.

[17] Urmila Mahadev. Classical verification of quantum computations. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 259–267. IEEE Computer Society, 2018.

[18] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[19] Chris Peikert and Sina Shiehian. Privately constraining and programming prfs, the lwe way. *IACR Cryptol. ePrint Arch.*, 2017:1094, 2018.

[20] Stefano Pirandola, Ulrik Lund Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, D. Englund, Tobias Gehring, Cosmo Lupo, Carlo Ottaviani, Jason L. Pereira, Mohsen Razavi, Jesni Shamsul Shaari, Marco Tomamichel, Vladyslav C. Usenko, Giuseppe Vallone, Paolo Villoresi, and Petros Wallden. Advances in quantum cryptography. *arXiv: Quantum Physics*, 2019.

[21] W. Quach, H. Wee, and D. Wichs. Laconic function evaluation and applications. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 859–870, Los Alamitos, CA, USA, oct 2018. IEEE Computer Society.

[22] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009.

[23] Or Sattath. Uncloneable Cryptography. 10 2022.

[24] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6), December 2015.

[25] Dominique Unruh. Collapse-binding quantum commitments without random oracles. In *Proceedings, Part II, of the 22nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032*, page 166–195, Berlin, Heidelberg, 2016. Springer-Verlag.

[26] Dominique Unruh. Computationally binding quantum commitments. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*, page 497–527, Berlin, Heidelberg, 2016. Springer-Verlag.

[27] A. C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, Oct 1986.

[28] J. Zhang. Classical verification of quantum computations in linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, Los Alamitos, CA, USA, nov 2022. IEEE Computer Society.

[29] Jiayu Zhang. *Succinct Blind Quantum Computation Using a Random Oracle*, page 1370–1383. Association for Computing Machinery, New York, NY, USA, 2021.

[30] Jiayu Zhang. Revisiting remote state preparation with verifiability: A new set of notions with well-behaved properties. 2023.