Software with Certified Deletion

James Bartusek[*] Vipul Goyal[†] Dakshita Khurana[‡] Giulio Malavolta[§]
Justin Raizes[¶] Bhaskar Roberts[||]

Abstract

Is it possible to prove the deletion of a computer program after having executed it? While this task is clearly impossible using classical information alone, the laws of quantum mechanics may admit a solution to this problem. In this work, we propose a new approach to answer this question, using quantum information. In the *interactive* settings, we present the first fully-secure solution for blind delegation with certified deletion, assuming post-quantum hardness of the learning with errors (LWE) problem. In the *non-interactive* settings, we propose a construction of obfuscation with certified deletion, assuming post-quantum iO and one-way functions.

Our main technical contribution is a new deletion theorem for subspace coset states [Vidick and Zhang, EUROCRYPT'21, Coladangelo et al., CRYPTO'21], which enables a generic compiler that adds the certified deletion guarantee to a variety of cryptographic primitives. In addition to our main result, this allows us to obtain a host of new primitives, such as functional encryption with certified deletion and secure software leasing for an interesting class of programs. In fact, we are able for the first time to achieve a *stronger notion of* secure software leasing, where even a *dishonest* evaluator cannot evaluate the program after returning it.

[*]UC Berkeley. Email: bartusek.james@gmail.com
[†]CMU and NTT Research. Email:vipul@cmu.edu
[‡]UIUC. Email: dakshita@illinois.edu
[§]Bocconi University & Max Planck Institute for Security and Privacy. Email: giulio.malavolta@hotmail.it
[¶]CMU. Email: jraizes@andrew.cmu.edu
[||]UC Berkeley. Email: bhaskarr@eecs.berkeley.edu

Contents

1 Introduction

Consider the following scenario: Alice is a software developer who has written a program that she would like to sell, in order to get financially rewarded for her effort. Bob is interested in using Alice's software, but only for a limited amount of time. Can Alice temporarily lease her software to Bob, without the risk of him pirating the program?

Ideally, we would like to design a protocol where Alice can lease her software to Bob and start charging him for a subscription fee. Once Bob is done using the software, he can produce a *deletion certificate* which guarantees that he deleted his local copy of the program. At this point, Alice can rest assured that Bob is no longer in possession of the software, and she can stop charging him. In case of a dispute, the deletion certificate will unequivocally determine which of the two parties misbehaved.

If we consider only classical information, then it is easy to see that no protocol can satisfy the security notion sketched above: Whatever information Alice sent to Bob, he can always create a perfect copy of it, thus continuing using the program even after producing the deletion certificate. In fact, nothing prevents Bob from pirating copies of Alice's program. On the other hand, the same argument does not hold if we consider quantum information, since the no-cloning theorem [WZ82] postulates that there does not exist a general algorithm to create perfect copies of quantum states. Indeed, recent works on quantum copy protection [Aar09] and secure software leasing [AL21, KNY21, BJL+21] propose using quantum information to solve similar problems. Unfortunately, all of the existing definitions are subject to strong impossibility results [AL21, AK22] and the aforementioned works either attain heuristic constructions in idealized models, or focus on restricted classes of programs. At present, the problem of general-purpose software with certified deletion is wide open (even with quantum information).

To better understand the challenge of constructing software with certified deletion, let us make more concrete the desiderata for such protocol. To cover the full spectrum of software with certified deletion, in this work we consider and formalize two complementary settings.

Interactive settings: Blind delegation with certified deletion. In the interactive settings, Alice is assisting Bob to evaluate the program, i.e., to compute the output of the software on some input, via an exchange of messages. This allows Alice to control exactly how many inputs Bob has queried to the software, so she can charge him accordingly.

It is well-known that *fully-homomorphic encryption* [Gen09] provides the ability to delegate a computation to an untrusted server, while revealing nothing about the computation itself. However, an FHE ciphertext is a classical string that information-theoretically contains Alice's software, and the only thing preventing recovery of this data is the conjectured hardness of a mathematical problem. If this problem becomes easy to solve in the future due to computational or scientific advances, or if Alice's secret key is leaked to Bob, there is no way to prevent Bob from recovering the underlying plaintext.

To mitigate this risk, we want Alice to be able to request Bob to delete their data at the end of the protocol, in such a way that, if Bob produces a valid certificate, then Alice is guaranteed that Bob has deleted the software *information theoretically*. This notion is known as *blind delegation with certified deletion*[1] [BI20, Por23, BK23]. However, existing proposals are actually insecure against a

[1] In the standard notion of blind delegation, Alice delegates the computation of a hidden input on a public function to Bob. Alice is then assumed to receive the output. However, this is easily seen to be equivalent to the version where

malicious Bob, who may deviate from the description of the protocol in an attempt to learn the Alice's software (more details on this later).

Non-interactive settings: Obfuscation with certified deletion. In the non-interactive settings, Alice ships a copy of the software to Bob, who can evaluate it freely on as many inputs as he wants. After some amount of time, Bob wants to produce a certificate of deletion that convinces Alice that, from that moment on, her software has been deleted information-theoretically. In other words, we want to encode a computer program into a quantum state that preserves its functionality, while enabling an evaluator to information-theoretically delete the underlying program.

We refer to this notion as *obfuscation with certified deletion*. Although a-priori it is not clear that this notion has anything to do with program obfuscation, we argue that the two are in fact intimately connected. After all, if Bob was able to learn Alice's software from its description, then there would really be no way to erase Bob's knowledge after the fact.

Limitations of existing approaches to certified deletion. Despite much recent progress designing cryptosystems with certified deletion [BI20, HMNY21, HMNY22b, Por23, BK23, HMNY22a], the above natural questions have remained unanswered. Arguably, we can trace the lack of progress back to the fact that we are missing a technique that allows for *repeated access* to partial information about the encoded data, followed by certified deletion of whatever is left. In other words, all works[2] thus far have focused on "all-or-nothing" style primitives: E.g. secret-key encryption, public-key encryption, attribute-based encryption, timed-release encryption, and commitments [Unr14, BI20, HMNY21, HMNY22b, BK23]. Even known constructions of *fully-homomorphic encryption* with certified deletion [Por23, BK23] are all-or-nothing, in the sense that security becomes compromised (as we show in this work) once we give the evaluator access to some type of decryption oracle.

1.1 Our Results

In this work, we introduce a new paradigm for secure information-theoretic deletion of data. Our main technical ingredient, that enables all of our results, is a new deletion theorem for *subspace coset states* [VZ21, CLLZ21]. Subspace coset states have previously been used for designing *uncloneable* cryptographic primitives [CLLZ21, AKL+22], and we demonstrate how to use them to obtain information-theoretic deletion. Our proof technique generalizes the work of [BK23] to states beyond BB84 states. This allows us to achieve *information-theoretic* certified deletion for cryptographic primitives beyond all-or-nothing. Specifically, we obtain the following results:

- **Blind delegation.** We develop the first maliciously secure blind delegation protocol with certified deletion. Our construction is based solely on the quantum hardness of learning

the function is also hidden, using universal circuits. Furthermore, we can let Bob receive the output by adding one more round.

[2] An exception to this claim is a very recent work of [HMNY22a] which constructs *functional encryption* with certified deletion of ciphertexts. However, in contrast with one of the goals of this work, their scheme is only secure in the setting of bounded collusions, where there is an a-priori upper bound on the number of functional keys an adversary can request.

with errors (LWE) [Reg05], and carefully combines subspace coset states with compact fully-homomorphic encryption (FHE) [Gen09] and succinct non-interactive arguments (SNARGs) for P.

We also provide the first construction of *two-message* blind delegation with certified deletion, based on post-quantum sub-exponentially secure indistinguishability obfuscation. In particular, once the client sends their encoding of x, the server can return *both* the evaluated output $f(x)$ *and* a certificate that all other information about x has been deleted without any more interaction with the client.

- **Obfuscation.** Assuming post-quantum indistinguishability obfuscation, we obtain the first construction of differing inputs obfuscation with certified deletion (di$\mathcal{O}$-CD), for a polynomial number of differing inputs. Loosely speaking, di$\mathcal{O}$-CD satisfies the standard notion of differing inputs obfuscation [BGI+12], in addition to the following certified deletion property: Let Π_0 and Π_1 two programs that differ on one input y^* (or a polynomial number of hard to find inputs), then it is hard to distinguish an obfuscation of Π_0 from an obfuscation of Π_1, *even given a differing input* y^*, provided that the distinguisher outputs the deletion certificate first. Intuitively, this formalizes the guarantee that, after deleting a program, one can no longer evaluate it on any input (more discussion on this later).

We can also conceptually abstract the above results as the following (informal) theorem, in an oracle model: For any classical functionality f, one can prepare an oracle that can be queried repeatedly (polynomially many times) before being *permanently deleted*. That is, after deletion, even an unbounded number of queries to the oracle will not reveal any more information about f. This general result may be of independent interest.

To demonstrate the usefulness of our newly developed tools, we show how they enable new applications in quantum cryptography, and in some cases they allow us to make progress on important open problems:

- **Secure Software Leasing.** As an immediate corollary of differing inputs obfuscation with certified deletion, we obtain a strong notion of secure software leasing for every differing inputs circuits family. Whereas the standard notion guarantees that the *honest* evaluation procedure fails for pirated copies of software, this strong notion guarantees security against arbitrary evaluation procedures.

- **Functional encryption.** We obtain two flavors of functional encryption with certified deletion: (i) one where *ciphertexts* can be certifiably deleted, and (ii) one where *secret keys* can be certifiably deleted (also known as key revocation or secure key leasing). The former assumes a strong-enough notion of post-quantum functional encryption (in particular, public-key multi-input FE with arity 2). The latter follows from differing inputs obfuscation with certified deletion, combined with post-quantum public-key encryption and injective one-way functions. Functional encryption with key revocation is our *only* result with a computational certified deletion guarantee. This is inherent in the primitive, as key revocation only emulates the case where a secret key was never received.

- **Public verification.** We develop a generic compiler that results in a variety of primitives with *publicly verifiable* certified deletion, assuming post-quantum indistinguishability obfuscation.

2 Technical Overview

2.1 Warm-Up Example

We illustrate the challenges and the techniques that we introduce in this work via a toy example. Namely, we will start from the, by now standard, notion of encryption with certified deletion and highlight the barriers that one encounters when trying to reveal some partial information about the plaintext. Specifically, we will try to build obfuscation with certified deletion starting from the latter.

Public-key encryption with certified deletion. We recall the basic notion of public-key encryption with certified deletion, and describe a recent construction due to [BK23] based on Wiesner encodings / BB84 states [Wie83, BB84]. For describing these states, we use the notation $|x\rangle_\theta$, where $x \in \{0,1\}^n$ is a string of bits, and $\theta \in \{0,1\}^n$ is a string of basis choices. Let Enc be the encryption algorithm for a post-quantum public-key encryption scheme. Then to encrypt a bit b, sample $x, \theta \leftarrow \{0,1\}^n$, and release

$$|x\rangle_\theta \, , \mathsf{Enc}(\theta, b \oplus \bigoplus_{i:\theta_i=0} x_i).$$

To delete, measure $|x\rangle_\theta$ in the Hadamard basis to obtain a string x'. This verifies as a valid deletion certificate if $x'_i = x_i$ for all $i : \theta_i = 1$. [BK23] show that since Enc is semantically secure and thus hides the choice of θ, any computationally-bounded adversary that produces a valid deletion certificate must have (essentially) measured most of the qubits in the Hadamard basis, erasing enough information about $\{x_i\}_{i:\theta_i=0}$ to claim that b is now statistically hidden.

Obfuscation and malleability. One nice property of the above scheme is that it can be decrypted classically after measuring $|x\rangle_\theta$ in the computational basis. This suggests a natural construction for obfuscation with certified deletion. First, encrypt (with certified deletion) the description of the circuit C. Then, obfuscate the classical program that does the following: given a circuit input, the secret key, and a classical measurement outcome obtained from the ciphertext encrypting C, recover the description of C, and then evaluate it on the input.

Unfortunately, such a construction *does not even satisfy indistinguishability obfuscation* (let alone any certified deletion guarantee). The issue is that the encryption scheme is clearly malleable: An adversary only has to guess a single index i where $\theta_i = 0$ in order to flip the message bit. Let's imagine an adversary that can maul the ciphertext to delete a single gate. It tries to distinguish an obfuscation of C_0 from one of C_1, where C_0 and C_1 are built from the same base circuit except that C_0 appends an identity gate and C_1 appends two consecutive NOT gates. This adversary can attempt to remove the last gate in the circuit. If this flips the output, the gate must have been C_1. This simple mauling capability therefore violates indistinguishability obfuscation (even *before* deletion). Under more sophisticated mauling attacks, it may even be possible to recover the whole circuit from the obfuscation!

Ciphertext validity check. A simple idea to overcome this issue is to enable the classically obfuscated program to check that the ciphertext has not been tampered with. Say the adversary

provides y to the obfuscated program as the alleged measurement of $|x\rangle_\theta$. To verify that the ciphertext is intact, the program only needs to verify that $y_i = x_i$ whenever $\theta_i = 0$. This can be done using a hard-coded x and θ. Otherwise, it can output $\perp$.

Unfortunately, the encryption scheme becomes completely insecure in the presence of such a program. An adversary can learn a description of θ one bit at a time, by flipping a bit of its state $|x\rangle_\theta$ and observing whether the program returns a successful evaluation or rejects. Once it learns θ, we cannot hope for any certified deletion guarantees. Moreover, the adversary can make additional queries to learn $\{x_i\}_{i:\theta_i=0}$, and, eventually, the circuit C.

Subspace coset states. Fortunately, there is a way to get around the problem that BB84 states are learnable in this sense. Prior work (for example, in the setting of publicly-verifiable quantum money) has switched to using entangled *subspace* states [AC12] and the more-general subspace coset states. A subspace coset state is defined by a subspace S of $\mathbb{F}_2^n$ and two vectors $\mathsf{v}, \mathsf{w} \in \mathbb{F}_2^n$, and is written as

$$|S_{\mathsf{v},\mathsf{w}}\rangle \coloneqq \frac{1}{\sqrt{|S|}} \sum_{\mathsf{z} \in S+\mathsf{v}} |\mathsf{z}\rangle \, (-1)^{\langle \mathsf{z},\mathsf{w}\rangle}.$$

It is useful to think of BB84 states as a type of subspace coset state in which the subspace is spanned by the standard basis vectors $\{e_i\}_{i:\theta_i=1}$. The coset in the primal space is determined by the bits $\{x_i\}_{i:\theta_i=0}$, which are used to hide the plaintext bit b, and the coset in the dual space is determined by the bits $\{x_i\}_{i:\theta_i=1}$, which determine what constitutes a valid deletion certificate.

Thus, in an attempt to make the obfuscation scheme secure, we replace the use of BB84 states with more-general subspace coset states. To encrypt each bit b of the description of the circuit, consider a ciphertext of the form

$$|S_{\mathsf{v},\mathsf{w}}\rangle \, , \mathsf{Enc}(S, C \oplus \langle \mathsf{v}, \mathbf{1} \rangle),$$

where we set S to be a random $n/2$-dimensional subspace, and a valid deletion certificate is now any vector $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$. The decryption algorithm, on input a vector z and ciphertext ct, will decrypt ct to obtain (S, b'), compute a canonical coset representative of $S + \mathsf{z}$, and use this resulting vector to unmask b.

Additionally, it is possible to check whether z has been tampered with by verifying that $\mathsf{z} \in S + \mathsf{v}$. It is even possible to publish an oracle for this consistency check, *without* leaking S and v [CLLZ21]. However, proving that the consistency oracle does not compromise the certified deletion security of the encryption scheme requires new ideas, and is a main technical contribution of this work.

Noisy consistency check. In order to carry out this consistency check, the obfuscated program must have S and v hard-coded. Unfortunately, the obfuscation only hides S and v computationally. After deletion, an unbounded adversary could learn v and $b \oplus \langle \mathsf{v}, \mathbf{1} \rangle$, which reveals b.

To information-theoretically protect b after deletion, we will instead sample a random superspace of S called T and hard-code the coset $T + \mathsf{u}$ that contains $S + \mathsf{v}$. We set $\dim(T) = 3n/4$ as a happy medium, which has two nice properties. First, since T is a negligible fraction of $\mathbb{F}_2^n$, it is hard for an adversary to find a vector in $T + \mathsf{u} \setminus S + \mathsf{v}$, so the consistency check will be essentially as good as using $S + \mathsf{v}$. Second, since S is a negligible fraction of T, $T + \mathsf{u}$ statistically hides enough information about v that $\langle \mathsf{v}, \mathbf{1} \rangle$ is uniformly random, even given $T + \mathsf{u}$. Therefore, the proof of certified deletion works.

It turns out that this "noisy consistency check" will be also be a crucial component in our constructions of both blind delegation and functional encryption with certified deletion.

2.2 General Compiler for Certified Deletion

Now we will present the tool that underlies all of our applications: a compiler that adds a certified deletion guarantee to a variety of cryptographic primitives.

First consider a simple template for certified deletion: to hide a bit b, we give the adversary the following state:

$$|S_{\mathsf{v},\mathsf{w}}\rangle\,, \mathcal{Z}(S, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle),$$

where $|S_{\mathsf{v},\mathsf{w}}\rangle$ is a random subspace coset state and $\mathcal{Z}$ is some side information, which may be classical. $\mathcal{Z}$ will often represent the primitive to which we are adding a certified deletion guarantee.

Note that given only the side information, b is statistically hidden because it is masked by $\langle \mathsf{v}, \mathbf{1}\rangle$. However, the information needed to remove the mask v is stored in the computational basis of the subspace coset state. To prove deletion, an honest party measures the subspace coset state in the Hadamard basis to get a vector $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$, which destroys essentially all information about v and removes b from their view. We will hope to prove that *any* strategy an (efficient) adversary uses to obtain a $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$ will also statistically remove b from their view.

The recent work of [BK23] showed how to prove this when $\mathcal{Z}$ satisfies *semantic security* with respect to S.[3] However our applications need a much richer set of choices for $\mathcal{Z}$ that do not necessarily hide S semantically. For instance, we want the ability to perform the noisy consistency check. That is: we want $\mathcal{Z}$ to output a randomized (T, u) where $S + \mathsf{v} \subset T + \mathsf{u}$. This is essential for our constructions of blind delegation and obfuscation with certified deletion. But $\mathcal{Z}$ would no longer hide S semantically because T reveals some basis vectors of $S^\perp$. In this case [BK23]'s proof falls short.

We present a compiler that supports a greater variety of choices for $\mathcal{Z}$, including the noisy consistency check. Specifically, we develop techniques to allow any choice of $\mathcal{Z}$ that satisfies a form of subspace-hiding against QPT adversaries. Morally, subspace-hiding means that an adversary cannot tell whether $\mathcal{Z}$ was testing membership in S (and $S^\perp$) or random superspaces $T \geq S$ (and $R \geq S^\perp$). We sketch our notion of subspace-hiding below:

Definition 2.1 (Subspace-Hiding, Informal)**.** *Given any subspace S of dimension $n/2$ and two cosets $S+\mathsf{v}$ and $S^\perp+\mathsf{w}$, let $T+\mathsf{u}$ and $R+\mathsf{x}$ be random cosets that contain the first two: $S+\mathsf{v} \subset T+\mathsf{u}$ and $S^\perp+\mathsf{w} \subset R+\mathsf{x}$. Then, $\mathcal{Z}(S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle)$ is subspace-hiding if there exists a simulator $\mathcal{S}(R, T, \mathsf{u}, \mathsf{x}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle)$ such that*

$$\mathcal{Z}(S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle) \approx_c \mathcal{S}(R, T, \mathsf{u}, \mathsf{x}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle),$$

where $\approx_c$ denotes indistinguishability to a quantum polynomial-time adversary.

Next, we claim that if $\mathcal{Z}$ satisfies subspace-hiding (which is a notion of *computational* security), then after the deletion certificate is accepted, b is *statistically* hidden, even if the inputs to $\mathcal{Z}$ are leaked at a later point. We sketch the security claim below.

Claim 2.2 (Certified Deletion Security, Informal)**.** *Let $\mathsf{EXP}(b)$ be the output of the following experiment:*

[3] Also they only consider the case where the quantum state is a string of BB84 states.

1. *Challenge: The challenger samples the following challenge and sends it to the adversary:*

$$|S_{\mathsf{v},\mathsf{w}}\rangle, \mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle)$$

2. *Response: The adversary responds with a deletion certificate $\widetilde{\mathsf{z}} \in \mathbb{F}_2^n$ and an auxiliary state ρ.*

3. *Outcome: The challenger checks that*

$$\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$$

If so, they output ρ and all the inputs to $\mathcal{Z}$; if not, they output $\perp$.

If $\mathcal{Z}$ is computationally subspace-hiding, then the statistical distance between $\mathsf{EXP}(0)$ and $\mathsf{EXP}(1)$ is negligible.

Proof overview. First, we claim that b is statistically hidden given *only* the side information $\mathcal{Z}(S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle)$. Note that b is masked by $\langle \mathsf{v}, \mathbf{1}\rangle$, and although $\mathcal{Z}$ may give some information about v in the form of (T, u), there is still some randomness left in v. In more detail, we can decompose v into its deterministic and random components by defining $\mathsf{v}_0 = \mathsf{v} - \mathsf{u}$. Then given $(S, T, \mathsf{u}, \mathsf{w})$, u is deterministic, and v_0 is uniformly random over $\mathsf{co}(S) \cap T$.[4] Because v_0 is uniformly random given $(S, T, \mathsf{u}, \mathsf{w})$, the bit $\langle \mathsf{v}, \mathbf{1}\rangle$ is also uniformly random (with overwhelming probability over the choice of (S, T)).

Next, recall that the adversary's view also includes the quantum state $|S_{\mathsf{v},\mathsf{w}}\rangle = |S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$, which stores v_0 in the computational basis. Now, b is not necessarily statistically hidden given both $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$ and $\mathcal{Z}(S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle)$. However, we will show that to prove deletion, the adversary must essentially measure $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$ in the Hadamard basis, destroying all information that the state had about v_0.

To show this, instead of giving the adversary $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$, we imagine giving them the following state, which stores a random $\widetilde{\mathsf{v}}_0$ in the Hadamard basis:

$$|T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle \quad \text{where } \widetilde{\mathsf{v}}_0 \leftarrow \mathsf{co}(T^\perp) \cap S^\perp.$$

$|T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle$ is in some sense dual with $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$. Both states store u in the computational basis and w in the Hadamard basis. The only difference is that $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$ encodes a random v_0 in the computational basis, and instead $|T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle$ encodes a random $\widetilde{\mathsf{v}}_0$ in the Hadamard basis. Furthermore, the adversary's behavior will be the same no matter which of the two states we give them. Indeed, we show that for any fixed $S, T, \mathsf{u}, \mathsf{w}$, the following states σ_0 and σ_1 are equivalent:[5]

$$\sigma_0 \propto \sum_{\mathsf{v}_0 \in \mathsf{co}(S) \cap T} |S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle \langle S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}|$$

$$\sigma_1 \propto \sum_{\widetilde{\mathsf{v}}_0 \in \mathsf{co}(T^\perp) \cap S^\perp} |T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle \langle T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}|$$

[4] $\mathsf{co}(S)$ is a group of coset representatives of S. See Section 5.1 for a precise definition of $\mathsf{co}(S)$.

[5] This is implicitly shown in Section 5.3 and Section 5.4 by purifying σ_0 and σ_1 and showing that there exists a unitary acting on the purifying register that maps between the two states.

This can be seen as a generalization of the fact that

$$\frac{1}{2}\big(\,|0\rangle\langle 0| + |1\rangle\langle 1|\,\big) = \frac{1}{2}\big(\,|+\rangle\langle +| + |-\rangle\langle -|\,\big).$$

In other words, the state is the same whether it's a maximal mixture of computational basis eigenstates or Hadamard basis eigenstates. Establishing this claim requires new techniques which seem to be generally useful for handling subspace coset states (more detail in Section 2.3).

Now, we want to argue that if the adversary produces a valid deletion certificate $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$, then given their remaining state, v_0 is statistically close to uniform. Imagine the adversary is given $|T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle$ and they output a valid deletion certificate $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$ with non-negligible probability. Recall that $T^\perp + \widetilde{\mathsf{v}}_0 + \mathsf{w}$ is an affine subspace of $S^\perp + \mathsf{w}$, so one way to do this is to make a measurement of the vector $\widetilde{\mathsf{v}}_0 + \mathsf{w}$ encoded in the phase, producing a vector in $T^\perp + \widetilde{\mathsf{v}}_0 + \mathsf{w}$. In fact, we show that since $\mathcal{Z}$ is computationally subspace-hiding for S, *any* adversary's strategy must be *statistically close* to making a measurement of $\widetilde{\mathsf{v}}_0 + \mathsf{w}$.[6] Then, if the adversary were instead given $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$, this same measurement of the phase would destroy all information about v_0, completing the proof.

2.3 Discussion

To gain some context, it is useful to zoom out from our main theorem, and compare our proof technique with existing works. As we shall see shortly, our settings require new proof techniques and cannot be framed as a special case of existing theorems.

New techniques for subspace coset states. While the previous section provides intuition, our actual proof is trickier and requires new techniques. Essentially, facts that are obvious for continuous vector spaces are sometimes false or difficult to formalize for discrete vector spaces. We develop new techniques for working with subspace cosets, and the culmination is an algorithm for *delayed preparation of subspace coset states*. Section 5 presents these results.

Our first contribution is to define a coset group $\mathsf{co}(S)$ that is isomorphic to $\mathbb{F}_2^n/S$ and that is a subspace of $\mathbb{F}_2^n$. This improves on prior work, [CLLZ21], which defined a set of canonical coset representatives that was not necessarily a group. The algebraic structure of $\mathsf{co}(S)$ allows us to prove more-sophisticated claims than what was possible with [CLLZ21]'s coset representatives.

Our second contribution is to develop a toolkit for proving such claims.

Our third contribution is an algorithm for delayed preparation of subspace coset states. This formalizes the intuition that the adversary's behavior is the same whether they are asked to play a game based on $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$ or a game based on $|T_{\mathsf{u},\tilde{\mathsf{v}}_0+\mathsf{w}}\rangle$. We formalize this by showing that we can purify σ_0 and σ_1 by adding a second register, and then show that there is a unitary that acts on the second register and maps the purification of σ_0 to that of σ_1. The unitary could be applied after the adversary acts on the first register, so the adversary's behavior will be the same in either case.

Why monogamy-of-entanglement techniques fail. Prior works [CLLZ21, Shm22] that dealt with subspace coset states relied on monogamy-of-entanglement (MoE) theorems, but these theorems fail to achieve the strong guarantees needed in our setting.

[6] That is, we use *computational* hardness to establish a *statistical* claim, as done in [BK23] in the context of BB84 states.

First, monogamy of entanglement is an information-theoretic property, and it does not necessarily hold if the adversary receives a computationally-secure encryption of the subspace S. We note that a recent work [AKL23] does use a MoE property to establish a certified deletion property, but crucially only in the information-theoretic one-time-pad encryption setting.

Next, MoE claims in prior work do not seem to easily extend to rule out the possibility that Bob outputs the string x, and Charlie simultaneously outputs the parity of x with non-negligible advantage. If this were possible, then even when one player produces a valid deletion certificate, the other player might learn a bit of data with non-negligible advantage, which would violate certified deletion security.

Why non-committing encryption techniques fail. Recall that we would like to eventually prove information-theoretic deletion of a secret that is initially information-theoretically determined by the adversary's view. Prior works (e.g., [AK21]) used receiver non-committing encryption schemes which have an "equivocality" property, allowing one to sample the fake keys after S is revealed. These were inherently limited to proving weaker forms of security; e.g., restricted to (computational) security against key-leakage attacks. Furthermore, equivocality is hard to achieve [KTZ13] for applications such as blind delegation, which involves FHE. Another setting where an equivocality-based approach fails is differing-inputs obfuscation. The choice of whether to behave as C_0 or C_1 is "hidden" under the differing inputs. Thus, the differing inputs act as a key to decrypt S, which reveals this choice bit. However, any differing input (i.e. key) y^* is easy to check by simply evaluating the two programs C_0 and C_1 on y^*, allowing fake keys to be immediately recognized. While [BK23] developed methods to overcome the equivocality issue for certified deletion, they only apply their techniques to settings where the subspace S is semantically hidden.

2.4 Blind Delegation with Certified Deletion

In this section, we discuss our construction of maliciously-secure blind delegation with certified deletion.

Insecurity of prior protocols against malicious adversaries. We first discuss why both prior protocols [Por23, BK23], while secure against semi-honest adversaries, are *insecure* against malicious adversaries.

Both of these protocols consist of four messages. First, the client encrypts their input m and sends a quantum ciphertext $|\mathsf{Enc}(m)\rangle$ to the server. Next, the server evaluates a function f to obtain a register holding a superposition over output ciphertexts $|\mathsf{Enc}(f(m))\rangle$, which is sent to the client. The client then coherently applies FHE decryption using their secret key, which allows them to recover $f(m)$ without disturbing the state, and then reverse their computation and send the undisturbed register back to the server. Finally, the server can uncompute f and recover the original ciphertext $|\mathsf{Enc}(m)\rangle$. Then, if they want, they can measure the ciphertext in a particular way to recover a certificate of deletion, which is sent to the client.

Now, consider the following attack. Suppose that the server wants to learn the first bit m_1 of m. They can easily prepare a state of the form

$$\frac{1}{\sqrt{2}}|\mathsf{Enc}(m_1)\rangle^{\mathsf{C}}|0\rangle^{\mathsf{S}} + \frac{1}{\sqrt{2}}|\mathsf{Enc}(0)\rangle^{\mathsf{C}}|1\rangle^{\mathsf{S}},$$

where $\mathsf{Enc}(0)$ is a freshly prepared encryption of 0. Then, suppose they send register C to the client in place of the second message of the protocol described above. In the case that $m_1 = 0$, the client's computation will not disturb the state, and the server will receive back the C register unharmed. But in the case when $m_1 = 1$, the client's measurement of the output *will* collapse the state. Thus, if the server unentangles register C and S, measures S in the Hadamard basis, and observes outcome $|-\rangle$, they will learn *for sure* that $m_1 = 1$, breaking privacy of the protocol.[7]

Our solution. The attack above relies on the fact that the client always immediately applies an operation that depends on their FHE secret key sk. To prevent this attack, we must introduce a way for the client to *check* that the server is honestly following the protocol, *before* using its secret key to operate on the state.

Suppose the client's input is a single bit b, and consider our basic encryption scheme

$$|S_{\mathsf{v},\mathsf{w}}\rangle\,, \mathsf{Enc}(S, b \oplus \langle \mathsf{v}, \mathbf{1}\rangle),$$

but where Enc is now instantiated as a fully-homomorphic encryption (FHE) scheme. We will have the server perform a classical FHE evaluation for circuit f in superposition over the vectors in $S + \mathsf{v}$, resulting in a superposition over $\mathsf{Enc}(f(b))$. Now, the client will need to perform two checks to make sure the server was behaving honestly:

- The client needs to check that the FHE evaluation in superposition was performed honestly. This can by accomplished by requesting that the server use a succinct non-interactive argument (SNARG) for P (polynomial-time computation) in superposition, and having the client verify this proof before decrypting. Moreover, SNARGs for P are known just from the LWE assumption [CJJ21], and it is straightforward to see that this SNARG remains post-quantum secure assuming the post-quantum hardness of LWE.

- The client *also* needs to check that the *input* to the server's computation is honest. This input is supposed to include any vector in $S + \mathsf{v}$. Thus, one solution is to have the client remember a description of $S + \mathsf{v}$, and also perform this check on the input before decrypting the output. However, this solution requires that the client keep the vector v around in memory, which if leaked would completely compromise certified deletion security. Instead, we use the same "noisy consistency check" as explained earlier, and have the client sample a random affine superspace $T + \mathsf{u}$ of $S + \mathsf{v}$, and only keep this around in memory.

This is the basic idea of our blind delegation scheme, and more details are given at the beginning of Section 7.2.

[7] This attack does not contradict any claims made in [Por23] or [BK23] because neither paper claims that their protocol is maliciously-secure. In [Por23], the correctness and security properties are defined entirely separately. That is, it is argued that correctness of the four-message protocol holds assuming parties are honest, but *security* (and certified deletion security) is only argued *assuming that the server does not interact with the client*. Thus, the claim is essentially that *either* correctness of delegation holds, *or* privacy against a malicious server holds. But it is never claimed that both can hold simultaneously. In [BK23], the authors do jointly consider correctness and security of the four-message protocol. However, they only claim security against servers that are *semi-honest* during the protocol execution (and potentially malicious after, while producing the deletion certificate). Thus, the above attack is explicitly disallowed by the semi-honest assumption.

Remarks on the security definition.

- We prove simulation-based security in the "protocols with certified deletion" framework of [BK23]. That is, we show that our protocol securely realizes an ideal functionality that takes input (f, x) from the client, delivers *only* f to the server, and delivers the output $f(x)$ to the client. Thus, in the simulated world, the server obtain *no* information about the client's private input x. We show that conditioned on the server producing a valid deletion certificate, their final view in the ideal protocol and in the ideal world are *statistically* indistinguishable, indicating that they have information-theoretically lost all information about x.
- We show that our protocol in fact realizes a *reusable* ideal functionality, where the client can request that the server compute for them multiple functions $f_1(x), f_2(x), \ldots$ on their original encrypted data x, and security still holds.
- In the case that deletion is accepted, we also explicitly leak all of the client's secret parameters to the adversary, and still require that statistical security holds. We capture this by defining a "long-term secrets" tape sec, where after each message, the honest client is supposed to write all of the information it needs to interact in the remainder of the protocol on this tape. Conditioned on deletion being accepted, we give the adversary access to this tape.

Finally, we remark that prior to our work, maliciously-secure blind delegation with certified deletion was not known *even without reusability*, and *even without requiring information-theoretic security after deletion*.

2.5 Obfuscation with Certified Deletion

A certifiably deletable program is an encoding of a classical circuit C into a quantum state $|\widetilde{C}\rangle$ that allows for evaluation of $C(x)$ on any input x. To realize certified deletion, there should also be a procedure for measuring $|\widetilde{C}\rangle$ in a particular way that certifiably destroys information about C.

While the natural notion of virtual black-box obfuscation is impossible to achieve in general [BGI+12], several relaxed notions are plausibly achievable. We focus on the case of differing inputs obfuscation for a polynomial number of differing inputs [BGI+12], which is implied by the related notion of indistinguishability obfuscation [BCP14]. This notion requires that for any two circuits C_0 and C_1 which differ on a polynomial number of hard-to-find inputs, an obfuscation of C_0 is indistinguishable from an obfuscation of C_1.

Our goal is thus to achieve differing inputs obfuscation with certified deletion for a polynomial number of differing inputs.[8] This is described by the following (simplified) game.

1. The challenger samples two programs C_0 and C_1 from a given distribution, where it holds that $C_0(y) = C_1(y)$, except for some y^*, that we refer to as the *differing input*. Importantly, even given the description of C_0 and C_1, it is computationally hard to find y^*.
2. The challenger flips a coin $b \leftarrow \{0, 1\}$ and sends an obfuscation of C_b to the attacker.
3. At some point of the experiment, the attacker outputs the deletion certificate, which is verified by the challenger.

[8] Our definition also generalizes to the case of an arbitrary number of differing inputs. However, achieving this would imply the existence of general differing inputs obfuscation, contrary to current evidence [BSW16, GGHW14].

4. If the certificate correctly verifies, then the challenger sends y^* to the attacker.

5. The attacker outputs a guess for the bit b.

We say that an obfuscation scheme is secure if the success probability of the attacker is negligibly close to $1/2$. We note that if the attacker becomes unbounded after outputting a valid deletion certificate, then they could compute y^* themselves. Thus, we remove Step 4 in our definition for information-theoretic certified deletion.

To justify this definition, we argue that it captures the intuitive guarantees that one would expect from software with certified deletion. First, there is a sense in which software with certified deletion implies some notion of obfuscation: if one could learn the program by just looking at its code, then there would be no point in issuing a deletion certificate. Second, we want to model the fact that once the adversary has deleted the program, they can no longer evaluate it on any input. However, it is not clear how to model the information which the adversary learned before deletion, i.e., when they had a *functional* copy of the program. We have no way of "looking inside the adversary's head" to learn which inputs they evaluate, and, even worse, the attacker may not even execute the program properly. Our definition sidesteps this issue, by requiring security for inputs that are hidden even given the plain description of the program (i.e., the differing inputs). In this sense, our definition can be interpreted as saying that the deletion prevents the adversary from learning any information that was not obviously leaked from having a running copy of the program.[9]

Construction. As outlined previously, the general structure of the construction is to encrypt the circuit C under a random coset v of the subspace S. Suppose for a moment that we can simultaneously hide all bits of C with a single vector v. To use the noisy consistency check, we will sample a uniform superspace $T + \mathsf{u}$ of $S + \mathsf{v}$. Then, we will hard-code S, T and u into a classical program $P[S, T, \mathsf{u}, C + \mathsf{v}]$ to be obfuscated. $P[S, T, \mathsf{u}, C + \mathsf{v}]$ takes as input a vector z (which should be in $S + \mathsf{v}$) and a string x to be evaluated. It checks that $z \in T + \mathsf{u}$, then computes the coset v of S that z belongs to, unmasks C using v, and finally computes and outputs $C(x)$. If $z \notin T + \mathsf{u}$, it aborts. Then, the construction is

$$|S_{\mathsf{v},\mathsf{w}}\rangle\,, \mathsf{Obf}(P[S, T, \mathsf{u}, C + \mathsf{v}])$$

To argue security, we would like to switch an obfuscation of C_0 to an obfuscation of C_1, and argue that this switch is *statistically* indistinguishable to an adversary that produces a successful deletion certificate. Our main theorem provides a way to obtain such statistical guarantees, but it only handles statistically hiding a single bit. Thus, we must perform a hybrid argument over the bits of the descriptions of the circuits. We cannot do this naively, since descriptions of circuits "in between" C_0 and C_1 are not guaranteed to be functionally equivalent to C_0 and C_1. Instead, we make use of the *two-slot technique* [NY01], and we defer details of this to the technical sections.

The above describes the main intuition and techniques that allow us to hide functionality, while still allowing for certified deletion. In the body of the paper, we also derive the following related results and applications.

[9] Our definition does not prevent an adversary from evaluating the program on *easy-to-find* inputs after deletion. This is because we cannot rule out the possibility of them having already evaluated those inputs before deletion.

Application: Strong Secure Software Leasing. Secure software leasing is defined with respect to a family of programs [AL21]. The adversary is given a leased program randomly chosen from this family and outputs two programs. If one of the programs is authenticated, then the other cannot be evaluated using the honest evaluation procedure.

We observe that any differing inputs program family can be securely leased by obfuscating it with certified deletion. A differing inputs program family contains pairs of programs (C_0, C_1) such that given a random pair, it is hard to find an input y^* where $C_0(y^*) \neq C_1(y^*)$. If an obfuscation of C_0 is returned to the lessor who then generates a valid deletion certificate, then the residual state cannot be used to distinguish whether the program was C_0 or C_1, even given a differing input y^*. In particular, the adversary that returned the program cannot later evaluate a pirated copy of it on y^* - otherwise they could check which program matched the output. Therefore, a leased program can be validated by attempting to delete it and checking the deletion certificate.

We emphasize that this guarantee is *stronger* than the original notion of secure software leasing, which permits the adversary to evaluate a pirated (i.e. unauthenticated) program as long as they do not use the honest evaluation procedure. In our definition, security is guaranteed even if the adversary uses an *arbitrary* evaluation procedure after returning a valid copy of the program.

Since we construct obfuscation with certified deletion for a polynomial number of differing inputs, we immediately obtain (strong) secure software leasing for differing inputs program families with a polynomial number of differing inputs. Existing impossibility results for secure software leasing [AL21, AK22] rule out secure software leasing for families containing programs which cannot be learned with black-box query access, but *can* be learned using non-black-box access to any functionally equivalent program. In contrast, a differing inputs program family contains programs which cannot be learned, even with non-black-box access to an obfuscation of the program.

Application: Functional Encryption with Key Revocation. To substantiate the usefulness of our definition, we show that our obfuscation scheme allows a simple and intuitive construction of public-key encryption, and even *functional encryption*, with key revocation. Moreover, our key revocation guarantee is *publicly-verifiable*. In key revocation, one or more secret keys are temporarily distributed to users. Later on, if the users comply with the revocation process, these keys are deleted and cannot be used to decrypt freshly generated ciphertexts [AKN+23, APV23].[10]

Our construction is essentially the same as the transformation from obfuscation to functional encryption given in [GGH+13], but our obfuscation scheme supports certified deletion. We describe a simplified version of their construction here. The secret key for a function f will be an obfuscated circuit that first decrypts a classical ciphertext to recover the message m, and then computes and returns $f(m)$. The encryption of m will use a standard public-key encryption scheme.

The above construction already guarantees that a key sk_f *only* reveals information about $f(m)$, by virtue of being a functional encryption scheme. The certified deletion property *additionally* ensures that, if the adversary has a key for f, but deletes it before receiving the challenge ciphertext, then he learns nothing. In fact, a straightforward reduction to the certified deletion security of the obfuscation scheme ensures that this is the case even if the adversary has access to other secret keys (security against *unbounded* collusion). We note that a similar technique allows adding publicly-verifiable key revocation to other encryption schemes, assuming iO.

[10] This property has also been referred to as secure key leasing.

3 Related Work

3.1 Prior Work

We first discuss prior works that build cryptographic schemes from subspace coset states. These states were first used by [VZ21] in the context of proofs of quantum knowledge and by [CLLZ21] to construct signature tokens (among other unclonable primitives) in the plain model. These were also used to build semi-quantum tokenized signatures [Shm22]. Most recently, [AKL+22] used subspace coset states to construct unclonable encryption satisfying the notion of unclonable indistinguishability. We remark that, while there are clearly similarities between the notions of unclonable encryption and encryption with certified deletion, our security definitions and proofs are quite different than those in [AKL+22]. For example, [AKL+22] crucially rely on *random oracles*, while our results are all in the plain model. Moreover, we achieve security definitions that promise *everlasting security* against unbounded adversaries after deletion, while [CLLZ21, AKL+22, Shm22] focuses on proving that computationally-bounded (or query-bounded) adversaries cannot perform a certain task, e.g., generating additional signatures.

Next, we mention two prior works that have considered functional encryption with certified deletion. [KN22] achieves a *private*-key version of functional encryption with certified deletion of secret keys. [HMNY22a] achieves functional encryption where the ciphertext can be deleted and it is certified everlasting (i.e. information theoretic certified deletion). Their construction is secure against bounded collusions, and either relies on the QROM or requires quantum certificates of deletion, and assumes public-key encryption. On the other hand, our functional encryption schemes support *public*-key encryption and are secure (in the sense of certified deletion) against an unbounded number of colluding users. We assume iO, which (up to subexponential hardness factors) is necessary, since it is implied by unbounded-collusion FE.

Finally, we remark that the blind delegation protocol of [Por23] is also shown to be publicly-verifiable, under the strong Gaussian-collapsing conjecture, and thus our results on blind delegation with public verification are technically incomparable. However, [Por23]'s conjecture involves an interactive game that has a baked in certified deletion component, wherein the adversary receives some trapdoor information conditioned on them successfully returning a pre-image of the hash function. On the other hand, indistinguishability obfuscation a priori has nothing to do with certified deletion. While post-quantum indistinguishability obfuscation is also not known from standard assumptions, this is a very active area of research with many candidates proposed over the last few years [BGMZ18, CVW18, BDGM22, GP21, WW21, DQV+21].

3.2 Concurrent and Independent Work

We will discuss three recent works [HKM+23, AKN+23, APV23] that construct revocable/deletable cryptographic primitives, a few of which overlap with ours. We compare the results in more detail below. At a high level, our constructions produce *publicly-verifiable, classical* deletion certificates in the plain model. The deletion certificates of [HKM+23, AKN+23, APV23] are all *privately* verifiable, and some of their constructions require *quantum* deletion certificates. Furthermore, our work unifies techniques using a general compiler for certified deletion based on subspace coset states. Whereas [HKM+23] and [AKN+23] developed different techniques for constructing, respectively, FE with certified deletion for ciphertexts and FE with key revocation, we construct both primitives from a single technique.

[HKM+23] construct functional encryption with certified deletion for *ciphertexts*. One of their main techniques is a way to verify BB84 states by individually signing them with a one-time signature. This serves a similar purpose as our use of subspace coset states. One difference is that their one-time signature approach results in privately-verifiable certificates of deletion, while our subspace coset state approach also supports public verification when combined with iO. They also construct several primitives with certified deletion that are not considered in our work: compute-and-compare obfuscation and predicate encryption. Likewise, we construct blind delegation, CCA encryption, and (differing-inputs) obfuscation, which they do not consider.

Next, [AKN+23] construct functional encryption that supports revocation of *secret keys*. They call this notion *secure key leasing*, which is the same as key revocation, except that their "certificate of deletion" is a privately-verifiable quantum state (the key itself) whereas ours is a classical string obtained by performing a destructive measurement on the quantum key. They also show how to add secure key leasing to various encryption schemes (public-key, identity-based, attribute-based, and functional) while requiring no additional assumptions. We only consider functional encryption with secure key leasing and assume iO, which is implied by subexponentially-secure functional encryption. Our technique is generic and could also be used to add publicly-verifiable secure key leasing to other encryption schemes, at the cost of assuming iO.

Finally, [APV23] also study revocable cryptography, which is the same notion of secure key leasing as studied by [AKN+23]. They obtain various primitives (including psuedorandom functions, PKE, and fully-homomorphic encryption) with key revocation, from the hardness of LWE. The comparison with our work is similar to the previous paragraph: [APV23] achieve constructions from standard assumptions, but only support privately-verifiable quantum certificates of revocation.

4 Preliminaries

Let λ denote the **security parameter**. A function f is **negligible** if for every constant $c \in \mathbb{N}$, $f(n) < n^{-c}$ for sufficiently large n. We use $\mathrm{negl}(n)$ to represent a generic negligible function of n. We say that an event happens with **overwhelming** probability if the probability is at least $1 - \mathrm{negl}(\lambda)$.

Let $\mathbb{F}_2$ denote the field over $\{0, 1\}$, where addition and multiplication are performed modulo 2. For any natural number n, $\mathbb{F}_2^n$ is a vector space. For any vectors $\mathsf{u}, \mathsf{v} \in \mathbb{F}_2^n$, let the **inner product** be computed mod 2: $\langle \mathsf{u}, \mathsf{v} \rangle = \sum_i \mathsf{u}_i \cdot \mathsf{v}_i \mod 2$. We denote the ith **standard basis vector** of $\mathbb{F}_2^n$ as e_i.

Given a vector $\mathsf{v} \in \mathbb{F}_2^n$: the **Hamming weight** is the number of nonzero entries in v. The **relative Hamming weight** $\omega(\mathsf{v})$ is the Hamming weight of v divided by n. The **parity** of v is denoted as $p(\mathsf{v})$ or $\langle \mathsf{v}, \mathbf{1} \rangle$.

4.1 Quantum Computation

A **quantum register** X is a set of n qubits, which represent the Hilbert space $\mathbb{C}^{2^n}$ and hold a quantum state. A **quantum state** on register X is represented as ρ or ρ^{X}, and it is a positive semi-definite operator on $\mathbb{C}^{2^n}$ with trace 1. We sometimes refer to this PSD operator as a **density matrix**. If the state is **pure**, it can also be represented as a unit vector $|\psi\rangle \in \mathbb{C}^{2^n}$.

A **quantum operation** F maps a state ρ on register X to a state σ on register Y. Formally, a quantum operation is a completely-positive trace-preserving (CPTP) map. Note that the two registers may have different sizes. Also, some possible notations for F include: $\sigma = F(\rho)$, $\sigma^{\mathsf{Y}} = F(\rho^{\mathsf{X}})$ and $\mathsf{Y} \leftarrow F(\mathsf{X})$.

A **unitary** $U : \mathsf{X} \to \mathsf{X}$ is a special case of a quantum operation that satisfies $U^\dagger U = UU^\dagger = \mathbb{I}^{\mathsf{X}}$, where $\mathbb{I}^{\mathsf{X}}$ is the identity matrix on register X. A **projector** Π is a Hermitian operator such that $\Pi^2 = \Pi$, and a **projective measurement** is a collection of projectors $\{\Pi_i\}_i$ such that $\sum_i \Pi_i = \mathbb{I}$.

Trace distance is the quantum analog of total variation distance: for two states ρ and σ, the trace distance between them is an upper bound on the advantage with which an (unbounded) algorithm can distinguish ρ and σ. Formally, the trace distance between ρ and σ is given by:

$$\mathsf{TD}(\rho, \sigma) := \frac{1}{2}\mathsf{Tr}\left(\sqrt{(\rho - \sigma)^\dagger(\rho - \sigma)}\right)$$

where $\mathsf{Tr}(\cdot)$ is the trace of the given matrix.

Lemma 4.1 (Gentle Measurement Lemma [Win99, CMSZ21])**.** *Let ρ be a quantum state in some Hilbert space, and let $\{\Pi, I - \Pi\}$ be a projective measurement that acts on that Hilbert space. Also, let (ρ, Π) satisfy: $\mathsf{Tr}(\Pi\rho) \geq 1 - \delta$.*

Next, let ρ' be the state that results from applying $\{\Pi, I - \Pi\}$ to ρ and post-selecting on obtaining the first outcome:

$$\rho' = \frac{\Pi\rho\Pi}{\mathsf{Tr}(\Pi\rho)}$$

Then $\mathsf{TD}(\rho, \rho') \leq 2\sqrt{\delta}$.

Finally, we'll define some common bases and states. For a single-qubit Hilbert space, the **computational basis** is $\{|0\rangle, |1\rangle\}$, and the **Hadamard basis** is $\{|+\rangle, |-\rangle\}$, where:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}},\ |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

The **Hadamard operation** H maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$.

The set of states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ are known as **BB84 states**. A **Wiesner state** is a string BB84 states. For strings $(x, \theta) \in \mathbb{F}_2^n$, the Wiesner state $|x\rangle_\theta$ is:

$$|x\rangle_\theta = H^{\theta_1}|x_1\rangle \otimes \cdots \otimes H^{\theta_n}|x_n\rangle$$

A **non-uniform quantum polynomial-time** (QPT) algorithm consists of an unitary U which can be implemented in polynomial time and a non-uniform quantum state $|\psi\rangle$. On input $|\phi\rangle$, it outputs $U(|\psi\rangle \otimes |\phi\rangle)$.

4.2 Subspaces and Cosets

A set $S \subseteq \mathbb{F}_2^n$ is a **subspace** if it is closed under vector addition and scalar multiplication. The notation $S \leq T$ means that S and T are subspaces, and $S \subseteq T$. For a subspace $S \leq \mathbb{F}_2^n$, the **orthogonal subspace** is $S^\perp = \{\mathsf{v} \in \mathbb{F}_2^n : \langle \mathsf{u}, \mathsf{v}\rangle = 0, \forall \mathsf{u} \in S\}$. Their dimensions satisfy $\dim(S) + \dim(S^\perp) = n$.

For any vector $\mathsf{z} \in \mathbb{F}_2^n$, the **coset** of S that z belongs to is the set: $S + \mathsf{z} = \{\mathsf{v} \in \mathbb{F}_2^n | \mathsf{v} = \mathsf{s} + \mathsf{z}, \exists \mathsf{s} \in S\}$. The cosets of S partition $\mathbb{F}_2^n$ into equal-sized sets: $|S + \mathsf{z}| = |S|$, and every z belongs to exactly one coset of S.

Finally, a **subspace coset state** is a state of the following form, where $\mathsf{v}, \mathsf{w} \in \mathbb{F}_2^n$:

$$|S_{\mathsf{v},\mathsf{w}}\rangle = \frac{1}{\sqrt{|S|}} \sum_{\mathsf{z} \in S + \mathsf{v}} |\mathsf{z}\rangle \cdot (-1)^{\langle \mathsf{z},\mathsf{w} \rangle}$$

Lemma 4.2 ([CLLZ21]). $H^{\otimes n} |S_{\mathsf{v},\mathsf{w}}\rangle = |S^{\perp}_{\mathsf{w},\mathsf{v}}\rangle$

Next, we will define a subspace $\mathsf{co}(S)$, which is analogous to the quotient group $\mathbb{F}_2^n / S$. $\mathsf{co}(S) \leq \mathbb{F}_2^n$ is a subspace that contains exactly one element of every coset of S. This is useful for decomposing vectors since every $\mathsf{z} \in \mathbb{F}_2^n$ has a *unique* decomposition as $\mathsf{z} = \mathsf{u} + \mathsf{v}$, for some $(\mathsf{u}, \mathsf{v}) \in S \times \mathsf{co}(S)$ (Lemma 5.1).

It is useful intuition to imagine choosing $\mathsf{co}(S) = S^{\perp}$, but this is not always allowed. Because S is a subspace of $\mathbb{F}_2^n$ and not $\mathbb{R}^n$, it's possible that some coset of S contains multiple vectors from $S^{\perp}$ and another coset contains none.

To avoid ambiguity, we always assume that a description of S includes a basis for each of the following subspaces: $[S, S^{\perp}, \mathsf{co}(S), \mathsf{co}(S^{\perp})]$.

See Section 5 for more details about $\mathsf{co}(S)$.

4.3 Obfuscation

4.3.1 Indistinguishability Obfuscation and Differing Inputs Obfuscation

Here we define three notions of obfuscation. First, indistinguishability obfuscation ($\mathsf{i}\mathcal{O}$) guarantees that for any two functionally equivalent circuits, their obfuscations are indistinguishable. Second, we define $\mathsf{i}\mathcal{O}$ for Turing machines, which is known as succint indistinguishability obfuscation. Third, we define differing inputs obfuscation ($\mathsf{di}\mathcal{O}$), which is similar to $\mathsf{i}\mathcal{O}$, but allows the two circuits to differ on a set of inputs as long as it is hard to find one of the differing inputs.

Definition 4.3 (Indistinguishability obfuscation)**.** *An indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is a PPT algorithm $\mathsf{i}\mathcal{O}$ that takes as input a security parameter 1^λ and a description of a circuit $C \in \mathcal{C}_\lambda$ and outputs an obfuscated circuit $\widetilde{C}$. It should satisfy the following properties.*

- ***Correctness.*** *For all $\lambda \in \mathbb{N}$, all $C \in \mathcal{C}_\lambda$, and all inputs x,*

$$\Pr[\mathsf{i}\mathcal{O}(1^\lambda, C)(x) = C(x)] = 1.$$

- ***Security.*** *For all sequences of functionally equivalent circuits $\{C_{0,\lambda}, C_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and all QPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$,*

$$|\Pr[\mathcal{A}_\lambda(\mathsf{i}\mathcal{O}(1^\lambda, C_{0,\lambda})) = 1] - \Pr[\mathcal{A}_\lambda(\mathsf{i}\mathcal{O}(1^\lambda, C_{1,\lambda})) = 1]| = \mathrm{negl}(\lambda).$$

Next, we define *succinct* indistinguishability obfuscation, otherwise known as $\mathsf{i}\mathcal{O}$ for Turing machines. We generally parameterize a Turing machine M with a step-size t and an input length n. Given M, t, and an $x \in \{0,1\}^n$, we define $M^t(x)$ to be the value written on the output tape of M after taking x as input and running for t steps, if such a value exists. Otherwise, it is defined to be $\perp$.

Definition 4.4 (Succinct indistinguishability obfuscation)**.** *A succinct indistinguishability obfuscator for a class of Turing machines $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is a PPT algorithm $\mathsf{si}\mathcal{O}$ that takes as input a security parameter 1^λ, the description of a Turing machine $M \in \mathcal{M}_\lambda$, a time bound t, and an input length n, and outputs an obfuscated Turing machine $\widetilde{M}$. It should satisfy the following properties.*

- ***Correctness.*** *For any $\lambda \in \mathbb{N}$, any $M \in \mathcal{M}_\lambda$, any $x \in \{0,1\}^n$, and any t,*

$$\Pr[\mathsf{si}\mathcal{O}(1^\lambda, M, t, n)(x) = M^t(x)] = 1.$$

- ***Security.*** *For all sequences of functionally equivalent Turing machines $\{M_{0,\lambda}, M_{1,\lambda}, t_\lambda, n_\lambda\}_{\lambda\in\mathbb{N}}$, meaning that for all $\lambda \in \mathbb{N}$, and $x \in \{0,1\}^{n(\lambda)}$, $M_{0,\lambda}^{t_\lambda}(x) = M_{1,\lambda}^{t_\lambda}(x)$, and all QPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$|\Pr[\mathcal{A}_\lambda(\mathsf{si}\mathcal{O}(1^\lambda, M_{0,\lambda}, t_\lambda, n_\lambda)) = 1] - \Pr[\mathcal{A}_\lambda(\mathsf{si}\mathcal{O}(1^\lambda, M_{1,\lambda}, t_\lambda, n_\lambda)) = 1]| = \mathrm{negl}(\lambda).$$

- ***Succinctness.*** *The running time of $\mathsf{si}\mathcal{O}$ must be $\mathrm{poly}(\lambda, |M|, \log t, n)$.*

Remark 4.5. *(Post-quantum) succinct indistinguishability obfuscation is known from (post-quantum) sub-exponentially secure indistinguishability obfuscation, one-way functions, and injective pseudo-random generators [KLW15].*

Differing inputs obfuscation is similar to indistinguishability obfuscation, but allows the two circuits to differ on a set of inputs as long as it is hard to find one of the differing inputs. We recall the definition of a differing input circuit family from [ABG+13].

Definition 4.6 (Differing Inputs Circuits)**.** *A circuit family $\mathcal{C}$ associated with an efficiently sampleable distribution $\mathcal{D}$ is said to be a differing input circuit family if for every PPT adversary $\mathcal{A}$,*

$$\Pr[C_0(x) \neq C_1(x) : (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}, x \leftarrow \mathcal{A}(1^\lambda, C_0, C_1, \mathsf{aux})] = \mathrm{negl}$$

If this holds against QPT adversaries, we say it is a post-quantum differing input circuit family. If C_0, C_1 differ on at most n inputs for all $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}$, we say $\mathcal{C}$ is a differing inputs circuit family with n differing inputs.

In this work, we only consider post-quantum differing input circuit families which differ on either 1 or polynomially many inputs. We emphasize that aux, which depends on the sampled circuits, is *classical*. A QPT adversary may additionally have non-uniform quantum advice $|\psi_\mathcal{A}\rangle$. This advice depends *only on the circuit family*, not on the sampled circuits, due to the order of the quantifiers. This definition could be strengthened by allowing aux to be quantum. However, a classical aux suffices for our applications.

Definition 4.7 (Differing Inputs Obfuscation)**.** *An indistinguishability obfuscator $\mathsf{di}\mathcal{O}$ for a differing inputs circuits family $\mathcal{C}$ associated with an efficiently sampleable distribution $\mathcal{D}$ over classical strings is a PPT or QPT algorithm such that:*

- ***Correctness:*** *For all circuits C, $\Pr[\tilde{C}(x) = C(x) \forall x : \tilde{C} \leftarrow \mathsf{di}\mathcal{O}(1^n, C)] = 1$*
- ***Indistinguishability:*** *For all differing inputs circuit classes and all PPT distinguishers D,*

$$\left| \begin{array}{c} \Pr[D(C_0, C_1, \mathsf{aux}, \tilde{C}) = 1 : \tilde{C} \leftarrow \mathsf{di}\mathcal{O}(1^n, C_0)), (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}] \\ -\Pr[D(C_0, C_1, \mathsf{aux}, \tilde{C}) = 1 : \tilde{C} \leftarrow \mathsf{di}\mathcal{O}(1^n, C_1), (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}] \end{array} \right| = \mathrm{negl}$$

If indistinguishability holds against QPT distinguishers, we say di$\mathcal{O}$ *is quantum-secure. If this holds and* di$\mathcal{O}$ *is a PPT algorithm, we instead say it is post-quantum.*

[BCP14] show that any indistinguishability obfuscator is also a differing inputs obfuscator for differing input circuits families with a polynomial number of differing inputs. Their ideas can be straightforwardly extended to show that this also holds for quantum adversaries, which consist of a unitary $U_{\mathcal{A}}$ and quantum advice $|\psi_{\mathcal{A}}\rangle$. Their proof performs a binary search over the input space by constructing intermediate obfuscations and asking the adversary to distinguish each of them. One may be concerned that performing a single distinguishing experiment may irreversibly collapse the adversary's internal state $|\psi_{\mathcal{A}}\rangle$, rendering it useless for subsequent experiments. However, if $|\psi_{\mathcal{A}}\rangle$ exists, then so does the state $|\psi_{\mathcal{A}}\rangle^{\otimes n}$, consisting of n copies of $|\psi_{\mathcal{A}}\rangle$. Since $|\psi_{\mathcal{A}}\rangle$ depends only on the circuit family (not the samples C_0, C_1), the copied state may be obtained non-uniformly.[11] Since aux is classical, it may be copied as well. Using one copy per distinguishing experiment allows the proof to succeed.

4.3.2 Subspace-Hiding Obfuscation

Next, we define the notion of *subspace-hiding* obfuscation, which provides a membership test for subspaces while hiding the subspace up to containment in a random superspace. Subspace-hiding obfuscation is implied by i$\mathcal{O}$ along with injective one-way functions.

First, let us define the membership test: given a set $S \subseteq \mathbb{F}_2^n$, let P_S be the program that takes as input a vector $\mathsf{v} \in \mathbb{F}_2^n$, and outputs 1 if $\mathsf{v} \in S$ and 0 otherwise. Second, when S is a subspace, we define a subspace-hiding obfuscator sh$\mathcal{O}$ to be a program that obfuscates P_S. For security, we say that an adversary cannot distinguish sh$\mathcal{O}(P_S)$ from sh$\mathcal{O}(P_T)$, where T is a random superspace of S.

Definition 4.8 ([Zha19], Def. 6.2)**.** *A subspace-hiding obfuscator* sh$\mathcal{O}$ *for a field* $\mathbb{F}$ *and dimensions* d_S, d_T *is a PPT algorithm* sh$\mathcal{O}$ *that takes a program* P_S *as input and outputs an obfuscated program* $\widetilde{P}$*.* sh$\mathcal{O}$ *is secure if all QPT adversaries have negligible advantage in the following game:*

- *The adversary sends the challenger a subspace* $S \leq \mathbb{F}^n$ *of dimension* d_S*.*
- *The challenger samples a random subspace* $T \leq \mathbb{F}^n$ *of dimension* d_T *such that* $S \leq T$*. Then they sample* $b \leftarrow \{0,1\}$*, and if* $b = 0$ *they compute* $\widetilde{P} \leftarrow$ sh$\mathcal{O}(P_S)$*, and if* $b = 1$*, they compute* $\widetilde{P} \leftarrow$ sh$\mathcal{O}(P_T)$*. Then they send* $\widetilde{P}$ *to the adversary.*
- *The adversary makes a guess* b' *for* b*.*

The adversary's advantage is $|\Pr[b' = b] - 1/2|$*.*

Theorem 4.9 ([Zha19], Theorem 6.3)**.** *If injective one-way functions exist and* $|\mathbb{F}|^{n-d_T}$ *is exponential, then any indistinguishability obfuscator, appropriately padded, is a subspace hiding obfuscator for field* $\mathbb{F}$ *and dimensions* d_S, d_T*.*

Next, we can extend the subspace-hiding guarantee to hold even when the program we're obfuscating checks membership in a coset of S.

[11] If the adversary is uniform, then its quantum auxiliary input state is empty. Therefore we may obtain multiple copies of it uniformly, and thus the reduction is uniform in this case.

Corollary 4.10 ([CLLZ21]). *Let $\mathsf{i}\mathcal{O}$ be an indistinguishability obfuscator and suppose that injective one-way functions exist, and consider the following game.*

- *The adversary sends the challenger a subspace $S < \mathbb{F}_2^n$ of dimension $n/2$ and a vector $\mathsf{v} \in \mathsf{co}(S)$.*
- *The challenger samples a random superspace $T > S$ of dimension $3n/4$ and defines $\mathsf{u} \in \mathsf{co}(T)$ to be the unique coset of T such that $S + \mathsf{v} \subseteq T + \mathsf{u}$. Then they sample $b \leftarrow \{0,1\}$. If $b = 0$, they compute $\widetilde{P} \leftarrow \mathsf{i}\mathcal{O}(P_{S+\mathsf{v}})$, and if $b = 1$ they compute $\widetilde{P} \leftarrow \mathsf{i}\mathcal{O}(P_{T+\mathsf{u}})$. Then they send $\widetilde{P}$ to the adversary.*
- *The adversary makes a guess b' for b.*

For any QPT adversary, it holds that $|\Pr[b' = b] - 1/2| = \mathrm{negl}(n)$ in the above game.

Proof. We consider a sequence of hybrids.

- $\mathcal{H}_0$: $\widetilde{P}$ is an obfuscation of $P_{S+\mathsf{v}}$.
- $\mathcal{H}_1$: Sample $\widetilde{P}_S \leftarrow \mathsf{i}\mathcal{O}(S)$ and let $\widetilde{P}$ be an obfuscation of the program $\widetilde{P}_S(\cdot - \mathsf{v})$ that takes as input a vector x and runs $\widetilde{P}_S(\mathsf{x} - \mathsf{v})$.
- $\mathcal{H}_2$: Sample $\widetilde{P}_T \leftarrow \mathsf{i}\mathcal{O}(T)$ and let $\widetilde{P}$ be an obfuscation of the program $\widetilde{P}_T(\cdot - \mathsf{v})$ that takes as input a vector x and runs $\widetilde{P}_T(\mathsf{x} - \mathsf{v})$.
- $\mathcal{H}_3$: $\widetilde{P}$ is an obfuscation of $P_{T+\mathsf{u}}$.

Indistinguishability between hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ and between hybrids $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from the security of $\mathsf{i}\mathcal{O}$. Indistinguishability between hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the security of $\mathsf{i}\mathcal{O}$ as a subspace-hiding obfuscator.

□

The following theorem is an information-theoretic version of Corollary 4.10. It doesn't assume $\mathsf{i}\mathcal{O}$, and instead, the adversary gets quantum query access to the function.

Theorem 4.11. *Consider the following game:*

1. *The adversary chooses a subspace $S \leq \mathbb{F}_2^n$ of dimension $n/2$ and a vector $\mathsf{v} \in \mathsf{co}(S)$. They send both to the challenger.*
2. *The challenger samples a random superspace $T > S$ of dimension $3n/4$ and defines $\mathsf{u} \in \mathsf{co}(T)$ to be the unique coset of T such that $S + \mathsf{v} \subseteq T + \mathsf{u}$. Then they sample $b \leftarrow \{0,1\}$ and if $b = 0$, they set $\widetilde{P} = P_{S+\mathsf{v}}$, and if $b = 1$ they set $\widetilde{P} = P_{T+\mathsf{u}}$. Then they give the adversary quantum query access to $\widetilde{P}$.*
3. *The adversary makes a guess b' for b.*

For any adversary that makes $\mathrm{poly}(n)$ queries, it holds that $\left|\Pr[b' = b] - 1/2\right| = \mathrm{negl}(n)$ in the game above.

Proof. We will use the inner-product adversary method of [AC12]. Without loss of generality, let the adversary's strategy after step 1 be the following: they start with a state $|\psi_0\rangle$ and then apply a sequence of unitaries and calls to the challenger's oracle $\widetilde{P}$. If $\widetilde{P} = P_{S+\mathsf{v}}$, their final state is $|\psi_{End}^{S+\mathsf{v}}\rangle$, and if $\widetilde{P} = P_{T+\mathsf{u}}$, let their final state be $|\psi_{End}^{T+\mathsf{u}}\rangle$. Finally, the adversary makes a single-qubit measurement on their state and outputs the result.

We will show that if $\left| \langle \psi_{End}^{S+\mathsf{v}} | \psi_{End}^{T+\mathsf{u}} \rangle \right| = 1 - \mathrm{negl}(n)$, this implies that the adversary's distinguishing advantage is $\mathrm{negl}(n)$. Let the adversary's distinguishing advantage be:

$$\mathsf{Advt} = \left| \Pr[b' = b] - 1/2 \right|$$

We can upper-bound Advt using the trace distance and fidelity between the final states $|\psi_{End}^{S+\mathsf{v}}\rangle$ and $|\psi_{End}^{T+\mathsf{u}}\rangle$:

$$\begin{aligned}
\mathsf{Advt} &= \left| \sum_T \Pr(T) \cdot [\Pr(b' = b|T) - 1/2] \right| \leq \sum_T \Pr(T) \cdot \left| \Pr(b' = b|T) - 1/2 \right| \\
&\leq \mathbb{E}_T \left[\mathsf{TD}\left(|\psi_{End}^{S+\mathsf{v}}\rangle \langle \psi_{End}^{S+\mathsf{v}}|, |\psi_{End}^{T+\mathsf{u}}\rangle \langle \psi_{End}^{T+\mathsf{u}}| \right) \right] \\
&\leq \mathbb{E}_T \left[\sqrt{1 - \left| \langle \psi_{End}^{S+\mathsf{v}} | \psi_{End}^{T+\mathsf{u}} \rangle \right|^2} \right] \leq \sqrt{1 - \mathbb{E}_T \left[\left| \langle \psi_{End}^{S+\mathsf{v}} | \psi_{End}^{T+\mathsf{u}} \rangle \right| \right]^2}
\end{aligned}$$

Next, the inner product $\langle \psi_{End}^{S+\mathsf{v}} | \psi_{End}^{T+\mathsf{u}} \rangle$ doesn't depend on the unitaries that were applied to the adversary's state; it only depends on the queries to $\widetilde{P}$. Let the progress measure p_t be the expected inner-product after t oracle queries:

$$p_t = \mathbb{E}_T \left[\left| \langle \psi_t^{S+\mathsf{v}} | \psi_t^{T+\mathsf{u}} \rangle \right| \right]$$

Now we will show that $|p_t - p_{t-1}| = \mathrm{negl}(n)$.

Note that for any $\mathsf{z} \in \mathbb{F}_2^n$, if $P_{T+\mathsf{u}}(\mathsf{z}) = 0$, then $P_{S+\mathsf{v}}(\mathsf{z}) = 0$. Next, if $P_{S+\mathsf{v}}(\mathsf{z}) = 0$, then

$$\Pr_{T+\mathsf{u}}[P_{T+\mathsf{u}}(\mathsf{z}) = 1] = \frac{|(T+\mathsf{u}) \backslash (S+\mathsf{v})|}{|\mathbb{F}_2^n \backslash (S+\mathsf{v})|} = \frac{2^{3n/4} - 2^{n/2}}{2^n - 2^{n/2}} = \frac{2^{n/4} - 1}{2^{n/2} - 1} < 2^{-n/4}$$

By Lemma 19 of [AC12],

$$|p_{t-1} - p_t| \leq 4\sqrt{2^{-n/4}} = \mathrm{negl}(n)$$

Finally, we know that $p_0 = 1$ because $|\psi_0^{S+\mathsf{v}}\rangle = |\psi_0^{T+\mathsf{u}}\rangle = |\psi_0\rangle$. Then after polynomially many queries, $p_t = 1 - \mathrm{negl}(n)$, so $\mathsf{Advt} = \mathrm{negl}(n)$. □

4.4 SNARGs for P

We define the notion of publicly-verifiable non-interactive delegation for a Turing machine $\mathcal{M}$. Such a scheme consists of algorithms $(\mathsf{SNARG.Gen}, \mathsf{SNARG.Prove}, \mathsf{SNARG.Verify})$ with the following syntax.

- $\mathsf{SNARG.Gen}(1^\lambda, \mathcal{M}, t, n) \to (\mathsf{pk}, \mathsf{vk})$ is an algorithm that takes as input a security parameter 1^λ, a Turing machine $\mathcal{M}$, a time bound t, and an input length n, and outputs a prover key pk and a verifier key vk.

- $\mathsf{SNARG.Prove}(\mathsf{pk}, x) \to (y, \pi)$ is an algorithm that takes as input a prover key pk and an input $x \in \{0,1\}^n$ and outputs $y \in \{0,1\}^m$ and a proof π.
- $\mathsf{SNARG.Verify}(\mathsf{vk}, x, y, \pi) \to \{0,1\}$ is an algorithm that takes as input a verifier key vk, an input x, an output y, and a proof π, and outputs either 0 or 1.

For Turing machine $\mathcal{M}$, let $\mathcal{U}_{\mathcal{M}}$ be the language

$$\mathcal{U}_{\mathcal{M}} \coloneqq \{(x, y, t) : \mathcal{M}(x) \text{ outputs } y \text{ within } t \text{ steps}\}.$$

Definition 4.12. *($\mathsf{SNARG.Gen}, \mathsf{SNARG.Prove}, \mathsf{SNARG.Verify}$) is a publicly-verifiable non-interactive delegation scheme for Turing machine $\mathcal{M}$ if the following hold.*

- ***Correctness:*** *For all λ, n, t such that $n \le t \le 2^\lambda$, and $(x, y, t) \in \mathcal{U}_{\mathcal{M}}$, it holds that*

$$\Pr\left[\mathsf{SNARG.Verify}(\mathsf{vk}, x, y, \pi) = 1 : \begin{array}{r}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{SNARG.Gen}(1^\lambda, \mathcal{M}, t, n) \\ (y, \pi) \leftarrow \mathsf{SNARG.Prove}(\mathsf{pk}, x)\end{array}\right] = 1.$$

- ***Efficiency:*** *In the above completeness experiment,* $\mathsf{SNARG.Prove}$ *runs in time* $\mathrm{poly}(\lambda, |\mathcal{M}|, t, n)$ *and* $\mathsf{SNARG.Verify}$ *runs in time* $\mathrm{poly}(\lambda, |\mathcal{M}|, \log t, n)$.
- ***Soundness:*** *For every QPT adversary $\{\mathcal{A}_\lambda\}_\lambda$ and polynomials $t(\lambda), n(\lambda)$, it holds that*

$$\Pr\left[\begin{array}{l}\mathsf{SNARG.Verify}(\mathsf{vk}, x, y, \pi) = 1 \\ (x, y, t) \notin \mathcal{U}_{\mathcal{M}}\end{array} : \begin{array}{r}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{SNARG.Gen}(1^\lambda, \mathcal{M}, t, n) \\ (x, y, \pi) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}, \mathsf{vk})\end{array}\right] = \mathrm{negl}(\lambda).$$

4.5 Fully-Homomorphic Encryption

A fully-homomorphic encryption scheme consists of algorithms $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ with the following syntax.

- $\mathsf{FHE.Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ is an algorithm that takes as input the security parameter and outputs a public key pk and a secret key sk.
- $\mathsf{FHE.Enc}(\mathsf{pk}, m) \to \mathsf{ct}$ is an algorithm that takes as input the public key pk and a message m, and outputs a ciphertext ct.
- $\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}) \to \widetilde{\mathsf{ct}}$ is an algorithm that takes as input the public key pk, a circuit C, and a ciphertext ct, and outputs an evaluated ciphertext $\widetilde{\mathsf{ct}}$.
- $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}) \to m$ is an algorithm that takes as input the secret key sk and a ciphertext ct, and outputs a message m.

An FHE scheme should satisfy **semantic security**, which states that the encryptions of two equal-length strings m_0, m_1 are computationally indistinguishable, and **correctness of evaluation**, which states that for any input m and circuit C, $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct})) = C(m)$, where $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, m)$. An FHE scheme should also satisfy some notion of **compactness**, which bounds the size of ciphertexts. In a levelled FHE scheme, the size of ct may grow with the depth of the circuit C to be evaluated, while in an unlevelled FHE scheme, the size of ct must be independent of C. Levelled FHE schemes are known from the hardness of the LWE, and unlevelled FHE schemes additionally require a circular-security assumption [Gen09, BV11, GSW13].

5 Delayed Preparation of Coset States

Here we develop tools for working with subspace coset states that will help us prove Theorem 6.5, which is our main theorem. In particular, we show how to prepare a random subspace coset state but delay the choice of subspace until after the register has been given out. Similar techniques exist for BB84/Wiesner states, but it is non-trivial to extend them to subspace coset states. Along the way, we develop a framework for representing the cosets of two subspaces $S \leq T$ that maintains the algebraic structure of the quotient groups $\mathbb{F}_2^n/S$ and $\mathbb{F}_2^n/T$. We believe the techniques in this section are interesting independently of their applications to certified deletion.

5.1 Coset Representatives

Given a subspace $S \leq \mathbb{F}_2^n$, let $\mathsf{co}(S)$ be a subspace of $\mathbb{F}_2^n$ that contains exactly one vector from every coset of S.[12] Note, $\mathsf{co}(S)$ is analogous to the quotient group $\mathbb{F}_2^n/S$, whose elements are the actual cosets of S and which has the same algebraic structure as $\mathsf{co}(S)$. $\mathsf{co}(S)$ is useful for decomposing vectors since every $\mathsf{z} \in \mathbb{F}_2^n$ has a *unique* decomposition as $\mathsf{z} = \mathsf{u} + \mathsf{v}$, for some $(\mathsf{u}, \mathsf{v}) \in S \times \mathsf{co}(S)$ (Lemma 5.1).

It is useful intuition to imagine choosing $\mathsf{co}(S) = S^\perp$, but this is not always allowed. Because S is a subspace of $\mathbb{F}_2^n$ and not $\mathbb{R}^n$, it's possible that some coset of S contains multiple vectors from $S^\perp$ and another coset contains none.

For any S, there exists a valid $\mathsf{co}(S)$ (see Definition 5.2 and Lemma 5.3). Usually, there are many valid choices of $\mathsf{co}(S)$, so we pick one of them to be canonical. To avoid ambiguity, let the description of S include a basis for the following subspaces: $[S, S^\perp, \mathsf{co}(S), \mathsf{co}(S^\perp)]$. This defines the canonical choices for $\mathsf{co}(S)$ and $\mathsf{co}(S^\perp)$.

Lemma 5.1 shows that the definition of $\mathsf{co}(S)$ is equivalent to some useful properties. In this lemma, we refer to $\mathsf{co}(S)$ as C.

Lemma 5.1. *For subspaces $S, C \leq \mathbb{F}_2^n$, the following are equivalent:*

1. *C contains exactly one element from each coset of S.*
2. *For any $\mathsf{z} \in \mathbb{F}_2^n$, there is a unique pair $(\mathsf{u}, \mathsf{v}) \in S \times C$ such that $\mathsf{z} = \mathsf{u} + \mathsf{v}$.*
3. *$\dim(C) = n - \dim(S)$ and $\mathsf{span}(S, C) = \mathbb{F}_2^n$.*

Proof. First, (1) implies (2). For any $\mathsf{z} \in \mathbb{F}_2^n$, z belongs to a coset of S, and there is a unique $\mathsf{v} \in C$ from the same coset. Let $\mathsf{u} = \mathsf{z} - \mathsf{v}$. Since z and v belong to the same coset, then $\mathsf{u} \in S$, and $\mathsf{z} = \mathsf{u} + \mathsf{v}$.

Second, (2) implies (3). (2) directly shows that $\mathsf{span}(S, C) = \mathbb{F}_2^n$. Next, we'll show that $\dim(C) = n - \dim(S)$. Every $\mathsf{z} \in \mathbb{F}_2^n$ has a *unique* decomposition in $S \times C$, so $|\mathbb{F}_2^n| = |S \times C|$. Therefore, $|C| = 2^n/|S|$, and $\dim(C) = n - \dim(S)$.

Third, (3) implies (1). $\mathbb{F}_2^n = \mathsf{span}(S, C)$, so for any vector $\mathsf{z} \in \mathbb{F}_2^n$, $\mathsf{z} = \mathsf{u} + \mathsf{v}$ for some $(\mathsf{u}, \mathsf{v}) \in S \times C$. Next, $\mathsf{v} = \mathsf{z} - \mathsf{u} \in S + \mathsf{z}$, so v is in both C and $S + \mathsf{z}$. Therefore, every coset shares at least one element with C. Furthermore, C contains exactly one vector from each coset because $|C| = 2^n/|S|$, which equals the number of cosets. □

[12] [CLLZ21] used a different set of coset representatives, called Can_S, which is not necessarily a vector space.

5.2 Sampling Procedure

In this section, we give a procedure for choosing $\mathsf{co}(S)$. We actually consider a more-general problem: given two subspaces $S \leq T$, we will choose $[\mathsf{co}(S), \mathsf{co}(S^\perp), \mathsf{co}(T), \mathsf{co}(T^\perp)]$ that satisfy $\mathsf{co}(T) \leq \mathsf{co}(S)$ along with some other useful properties. In later sections, whenever we need to sample two subspaces, $S \leq T$, we will implicitly use Definition 5.2 to sample the associated coset representatives.

Definition 5.2 (Procedure to Sample Coset Representatives)**.** *Given two subspaces $S \leq T \leq \mathbb{F}_2^n$:*

1. *Choose n linearly independent vectors $\{\mathsf{z}_1, \ldots, \mathsf{z}_n\}$ uniformly at random such that*

$$S = \mathsf{span}(\mathsf{z}_1, \ldots, \mathsf{z}_{\dim(S)})$$
$$T = \mathsf{span}(\mathsf{z}_1, \ldots, \mathsf{z}_{\dim(T)})$$

2. *Then let*

$$\mathsf{co}(S) = \mathsf{span}(\mathsf{z}_{\dim(S)+1}, \ldots, \mathsf{z}_n)$$
$$\mathsf{co}(T) = \mathsf{span}(\mathsf{z}_{\dim(T)+1}, \ldots, \mathsf{z}_n)$$
$$\mathsf{co}(S^\perp) = \mathsf{co}(S)^\perp$$
$$\mathsf{co}(T^\perp) = \mathsf{co}(T)^\perp$$

3. *Choose a fresh random basis for each subspace in $[\mathsf{co}(S), \mathsf{co}(S^\perp), \mathsf{co}(T), \mathsf{co}(T^\perp)]$, and output these bases. Note that the subspaces do not change in this step – just the bases used to represent them.*

The reason we choose fresh random bases for each subspace is so that someone with a description of S and $\mathsf{co}(S)$ but not T does not learn anything about T other than the fact that $S \leq T$. The original basis we chose for $\mathsf{co}(S)$ was built from the basis for T, which might leak information about T.

Lemma 5.3 analyzes the procedure in Definition 5.2 and proves that it satisfies some useful properties.

Lemma 5.3. *Given two subspaces $S \leq T \leq \mathbb{F}_2^n$, the procedure in Definition 5.2 chooses the subspaces $[\mathsf{co}(S), \mathsf{co}(S^\perp), \mathsf{co}(T), \mathsf{co}(T^\perp)]$ such that:*

1. *$\mathsf{co}(S)$ is valid: it contains exactly one element from each coset of S. The analogous statement holds for $\mathsf{co}(S^\perp), \mathsf{co}(T), \mathsf{co}(T^\perp)$.*
2. *$\mathsf{co}(T) \leq \mathsf{co}(S)$ and $\mathsf{co}(S^\perp) \leq \mathsf{co}(T^\perp)$.*
3. *$\mathsf{co}(S^\perp) = \mathsf{co}(S)^\perp$ and $\mathsf{co}(T^\perp) = \mathsf{co}(T)^\perp$.*

Proof. To prove property 1, we will show that $\dim[\mathsf{co}(S)] = n - \dim(S)$ and $\mathsf{span}[S, \mathsf{co}(S)] = \mathbb{F}_2^n$. By Lemma 5.1, this implies that $\mathsf{co}(S)$ contains exactly one element from each coset of S. We will show the analogous statements for $\mathsf{co}(S^\perp), \mathsf{co}(T), \mathsf{co}(T^\perp)$.

First, it is clear by the construction of $\mathsf{co}(S)$ that $\dim[\mathsf{co}(S)] = n - \dim(S)$. It is also clear that $\dim[\mathsf{co}(T)] = n - \dim(T)$. Next,

$$\dim[\mathsf{co}(S^\perp)] = \dim[\mathsf{co}(S)^\perp] = n - \dim[\mathsf{co}(S)] = n - [n - \dim(S)] = \dim(S) = n - \dim(S^\perp)$$

By the same argument, we can show that $\dim[\mathsf{co}(T^\perp)] = n - \dim(T^\perp)$.

Second,

$$\mathsf{span}[S, \mathsf{co}(S)] = \mathsf{span}(\mathsf{z}_1, \ldots, \mathsf{z}_n) = \mathbb{F}_2^n$$

because the vectors $\{\mathsf{z}_i\}_{i\in[n]}$ are linearly independent. By the same reasoning, we can show that $\mathsf{span}[T, \mathsf{co}(T)] = \mathbb{F}_2^n$.

Next, we will prove that $\mathsf{span}[S^\perp, \mathsf{co}(S)^\perp] = \mathbb{F}_2^n$. First, $S \cap \mathsf{co}(S) = \{\mathbf{0}\}$. This is because $\mathsf{co}(S)$ contains exactly one element of S (Lemma 5.1), and the shared element has to be $\mathbf{0}$ because S and $\mathsf{co}(S)$ are both subspaces. Second:

$$\mathbb{F}_2^n = \{\mathbf{0}\}^\perp = [S \cap \mathsf{co}(S)]^\perp = \mathsf{span}[S^\perp, \mathsf{co}(S)^\perp]$$

For the same reason, $\mathsf{span}[T^\perp, \mathsf{co}(T)^\perp] = \mathbb{F}_2^n$.

Now we'll prove property 2. It is clear from the construction that $\mathsf{co}(T) \leq \mathsf{co}(S)$. Next,

$$\mathsf{co}(S^\perp) = \mathsf{co}(S)^\perp \leq \mathsf{co}(T)^\perp = \mathsf{co}(T^\perp)$$

Finally, property 3 is clearly true from the construction. □

5.3 Delayed Preparation of Coset States

Our goal in this section is for Alice to prepare a random subspace coset state for Bob, but delay choosing the underlying subspace until after she sends the register to Bob. This technique is used in the proof of the main theorem, Theorem 6.5, and it uses the formalism for coset representatives that we developed above.

Let Alice be given two subspaces $S \leq T \leq \mathbb{F}_2^n$, and let the corresponding coset representatives $[\mathsf{co}(S), \mathsf{co}(S^\perp), \mathsf{co}(T), \mathsf{co}(T^\perp)]$ be sampled from the procedure in Definition 5.2. Next, let Alice be given cosets $\mathsf{u} \in \mathsf{co}(T)$ and $\mathsf{w} \in \mathsf{co}(S^\perp)$, which partially determine the subspace coset state that Alice will sample.

Alice will sample the subspace coset state from one of the following distributions:

- Distribution 0: Sample $|S_{\mathsf{v},\mathsf{w}}\rangle$ such that $S + \mathsf{v} \subseteq T + \mathsf{u}$, uniformly at random.
- Distribution 1: Sample $|T_{\mathsf{u},\widetilde{\mathsf{v}}}\rangle$ such that $T^\perp + \widetilde{\mathsf{v}} \subseteq S^\perp + \mathsf{w}$, uniformly at random.

Bob's register will eventually contain the sampled state. But there's a twist: Alice will decide which distribution to sample from *after* she sends Bob her register.

Here is one way for Alice to sample from distribution 0 or 1:

1. To sample from distribution 0, Alice prepares the following state on two n-qubit registers:

$$|\psi\rangle_0 := \frac{1}{\sqrt{|A|}} \sum_{\mathsf{v}_0 \in A} |S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle\, |\mathsf{v}_0\rangle$$

   where $A = \mathsf{co}(S) \cap T$. She sends the first register to Bob, and measures the second register in the computational basis.

2. To sample from distribution 1, she prepares the following state:

$$|\psi\rangle_1 := \frac{1}{\sqrt{|B|}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle \, |\widetilde{\mathsf{v}}_0\rangle$$

where $B = \mathsf{co}(T^\perp) \cap S^\perp$. She sends the first register to Bob, and measures the second register in the computational basis.

Claim 5.4. *The procedure above correctly samples from distribution 0 or distribution 1.*

Proof. In this procedure, we have decomposed v into its deterministic and random components, u and v_0 respectively. Every $\mathsf{v} \in \mathsf{co}(S)$ has a unique decomposition as $\mathsf{v} = \mathsf{u} + \mathsf{v}_0$ for some $\mathsf{u} \in \mathsf{co}(T)$ and $\mathsf{v}_0 \in A$ (Lemma 5.5). Since v_0 is sampled uniformly at random from A, v is sampled uniformly at random such that $\mathsf{v} \in \mathsf{co}(S)$ and $S + \mathsf{v} \subseteq T + \mathsf{u}$. Therefore, the procedure correctly samples from distribution 0.

A similar argument works for distribution 1. We have decomposed $\widetilde{\mathsf{v}}$ into its deterministic and random components, w and $\widetilde{\mathsf{v}}_0$ respectively. Every $\widetilde{\mathsf{v}} \in \mathsf{co}(T^\perp)$ has a unique decomposition as $\widetilde{\mathsf{v}} = \widetilde{\mathsf{v}}_0 + \mathsf{w}$ for some $\widetilde{\mathsf{v}}_0 \in B$ and $\mathsf{w} \in \mathsf{co}(S^\perp)$ (Lemma 5.6). Since $\widetilde{\mathsf{v}}_0$ is sampled uniformly at random from B, $\widetilde{\mathsf{v}}$ is sampled uniformly at random such that $\widetilde{\mathsf{v}} \in \mathsf{co}(T^\perp)$ and $T^\perp + \widetilde{\mathsf{v}} \subseteq S^\perp + \mathsf{w}$. Therefore, the procedure correctly samples from distribution 1. □

Next, Alice can map between $|\psi\rangle_0$ and $|\psi\rangle_1$ by applying local operations to the second register. We will define a unitary U that acts on the second register and maps superpositions over to A to superpositions over B. For any $\mathsf{v}_0 \in A$, let

$$U \, |\mathsf{v}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |\widetilde{\mathsf{v}}_0\rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0 \rangle}$$

and for any $\widetilde{\mathsf{v}}_0 \in B$, let

$$U^\dagger \, |\widetilde{\mathsf{v}}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\mathsf{v}_0 \in A} |\mathsf{v}_0\rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0 \rangle}$$

where c is a normalization constant. Technically, U acts on any superposition over $\mathbb{F}_2^n$, and we define it fully in Definition 5.8. We show in Lemma 5.13 that when U is applied to the second register of $|\psi\rangle_0$, it maps the state to $|\psi\rangle_1$.

Now we have the tools to do delayed preparation of the subspace coset state:

Delayed Preparation of a Subspace Coset State

1. Alice prepares $|\psi\rangle_0$ on two n-qubit registers. She sends the first register to Bob.
2. (a) To sample from distribution 0: Alice measures the second register in the computational basis to get a random $\mathsf{v}_0 \leftarrow A$. The state on Bob's register collapses to $|S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle$.
   (b) Instead, to sample from distribution 1: Alice applies U to the second register, mapping $|\psi\rangle_0$ to $|\psi\rangle_1$. Then she measures the second register in the computational basis to get a random $\widetilde{\mathsf{v}}_0 \leftarrow B$. The state on Bob's register collapses to $|T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle$.

5.4 Lemmas

This section provides the lemmas that were used in Section 5.3 to develop the protocol for delayed preparation of subspace coset states.

Recall that for a given $S \leq T$: $A = \mathsf{co}(S) \cap T$ and $B = \mathsf{co}(T^\perp) \cap S^\perp$.

Lemma 5.5 (Decomposition into $S \times A \times \mathsf{co}(T)$)**.** *For each* $\mathsf{z} \in \mathbb{F}_2^n$*, there is a unique tuple* $(\mathsf{s}, \mathsf{v}_0, \mathsf{u}) \in S \times A \times \mathsf{co}(T)$ *such that* $\mathsf{z} = \mathsf{s} + \mathsf{v}_0 + \mathsf{u}$*. Also* $\mathsf{span}(S, A) = T$*, and* $\mathsf{span}[A, \mathsf{co}(T)] = \mathsf{co}(S)$*.*

Proof. For every $\mathsf{z} \in \mathbb{F}_2^n$, there is a unique pair $(\mathsf{t}, \mathsf{u}) \in T \times \mathsf{co}(T)$ such that $\mathsf{z} = \mathsf{t} + \mathsf{u}$ (Lemma 5.1). Furthermore, t can be decomposed as $\mathsf{t} = \mathsf{s} + \mathsf{v}_0$ for a unique pair $(\mathsf{s}, \mathsf{v}_0) \in S \times \mathsf{co}(S)$.

Next, $\mathsf{v}_0 \in A$. To see this, note that $\mathsf{v}_0 = \mathsf{t} - \mathsf{s} \in \mathsf{span}(T, S) = T$, and $\mathsf{v}_0 \in \mathsf{co}(S)$. Therefore, $\mathsf{v}_0 \in \mathsf{co}(S) \cap T = A$.

Now we'll show that $\mathsf{span}(S, A) = T$. This is because $S, A \leq T$, and each $\mathsf{t} \in T$ can be decomposed as $\mathsf{t} = \mathsf{s} + \mathsf{v}_0$ for some $(\mathsf{s}, \mathsf{v}_0) \in S \times A$.

Finally, we'll show that $\mathsf{span}[A, \mathsf{co}(T)] = \mathsf{co}(S)$. First, $A, \mathsf{co}(T) \leq \mathsf{co}(S)$ (Lemma 5.3). Next, each $\mathsf{z} \in \mathsf{co}(S)$ can be decomposed as $\mathsf{z} = \mathsf{s} + \mathsf{v}_0 + \mathsf{u}$ for a unique $(\mathsf{s}, \mathsf{v}_0, \mathsf{u}) \in S \times A \times \mathsf{co}(T)$. Furthermore, $\mathsf{s} = \mathbf{0}$ because otherwise $\mathsf{co}(S)$ would contain two vectors in the same coset of S: z and $\mathsf{z} - \mathsf{s}$. Therefore, $\mathsf{z} \in \mathsf{span}[A, \mathsf{co}(T)]$, so $\mathsf{co}(S) \subseteq \mathsf{span}[A, \mathsf{co}(T)]$. □

Lemma 5.6 (Decomposition into $T^\perp \times B \times \mathsf{co}(S^\perp)$)**.** *For each* $\mathsf{z} \in \mathbb{F}_2^n$*, there is a unique tuple* $(\mathsf{t}, \widetilde{\mathsf{v}}_0, \mathsf{w}) \in T^\perp \times B \times \mathsf{co}(S^\perp)$ *such that* $\mathsf{z} = \mathsf{t} + \widetilde{\mathsf{v}}_0 + \mathsf{w}$*. Also* $\mathsf{span}(T^\perp, B) = S^\perp$*, and* $\mathsf{span}[B, \mathsf{co}(S^\perp)] = \mathsf{co}(T^\perp)$*.*

Proof. The proof is analogous to that of Lemma 5.5. □

Next, we'll define the function U and prove that it is unitary.

Lemma 5.7. $|A| = |B| = |T|/|S|$

Proof. S is a subgroup of T, so T can be partitioned into $|T|/|S|$ cosets of S. Therefore, $|A| = |\mathsf{co}(S) \cap T| = |T|/|S|$.

Next, $T^\perp \leq S^\perp$ because $S \leq T$. Therefore $S^\perp$ can be partitioned into $|S^\perp|/|T^\perp|$ cosets of $T^\perp$. Finally,

$$|B| = |\mathsf{co}(T^\perp) \cap S^\perp| = \frac{|S^\perp|}{|T^\perp|} = \frac{2^n}{|S|} \cdot \frac{|T|}{2^n} = \frac{|T|}{|S|}$$

□

Definition 5.8 (Unitary U)**.**

- *Let* $c = |T|/|S| = |A| = |B|$ *(see Lemma 5.7)*
- *Let* $\overline{A} = \mathbb{F}_2^n \backslash A$*, and* $\overline{B} = \mathbb{F}_2^n \backslash B$*.*
- *Let* f *be a bijective mapping from* $\overline{A}$ *to* $\overline{B}$*. Such an* f *exists because* $\overline{A}$ *and* $\overline{B}$ *have the same size.*
- *Let* U *be the linear function mapping any* $\mathsf{v}_0 \in A$ *to*

$$U\,|\mathsf{v}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |\widetilde{\mathsf{v}}_0\rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0 \rangle}$$

and any $\mathsf{v}_0 \in \overline{A}$ *to* $|f(\mathsf{v}_0)\rangle$*.*

Lemma 5.9. *$U^\dagger$ is the linear function mapping any $\widetilde{\mathsf{v}}_0 \in B$ to*

$$U^\dagger |\widetilde{\mathsf{v}}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\mathsf{v}_0 \in A} |\mathsf{v}_0\rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0 \rangle}$$

and any $\widetilde{\mathsf{v}}_0 \in \overline{B}$ to $|f^{-1}(\widetilde{\mathsf{v}}_0)\rangle$.

Proof. For any $\mathsf{v}_0 \in A$ and any $\widetilde{\mathsf{v}}_0 \in B$,

$$\langle \widetilde{\mathsf{v}}_0 | U |\mathsf{v}_0\rangle = \frac{1}{\sqrt{c}} \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0 \rangle}$$

If $\mathsf{v}_0 \notin A$ or $\widetilde{\mathsf{v}}_0 \notin B$, then

$$\langle \widetilde{\mathsf{v}}_0 | U |\mathsf{v}_0\rangle = \mathbb{1}_{\widetilde{\mathsf{v}}_0 == f(\mathsf{v}_0)}$$

□

Lemma 5.10. *U is unitary.*

Proof. To prove the claim, we will show that for any $\mathsf{v}_0, \mathsf{v}'_0 \in \mathbb{F}_2^n$, $\langle \mathsf{v}'_0 | U^\dagger U |\mathsf{v}_0\rangle = \mathbb{1}_{\mathsf{v}_0 == \mathsf{v}'_0}$.

First, if v_0 or v'_0 are not in A, then

$$\langle \mathsf{v}'_0 | U^\dagger U |\mathsf{v}_0\rangle = \mathbb{1}_{\mathsf{v}'_0 == f^{-1} \circ f(\mathsf{v}_0)} = \mathbb{1}_{\mathsf{v}_0 == \mathsf{v}'_0}$$

Now let $\mathsf{v}_0, \mathsf{v}'_0 \in A$.

$$\begin{aligned}\langle \mathsf{v}'_0 | U^\dagger U |\mathsf{v}_0\rangle &= \frac{1}{c} \sum_{\widetilde{\mathsf{v}}'_0 \in B} \sum_{\widetilde{\mathsf{v}}_0 \in B} \langle \widetilde{\mathsf{v}}'_0 | \widetilde{\mathsf{v}}_0 \rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0 \rangle - \langle \mathsf{v}'_0, \widetilde{\mathsf{v}}'_0 \rangle} \\ &= \frac{1}{c} \sum_{\widetilde{\mathsf{v}}_0 \in B} (-1)^{\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle}\end{aligned}$$

If $\mathsf{v}_0 = \mathsf{v}'_0$, then $\langle \mathsf{v}'_0 | U^\dagger U |\mathsf{v}_0\rangle = \frac{1}{c} \sum_{\widetilde{\mathsf{v}}_0 \in B} 1 = 1$.

Now let $\mathsf{v}_0 - \mathsf{v}'_0 \neq 0$. Lemma 5.11 says that for half of the vectors $\widetilde{\mathsf{v}}_0 \in B$, $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle = 1$, and for the other half, $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle = 0$. Therefore,

$$\langle \mathsf{v}'_0 | U^\dagger U |\mathsf{v}_0\rangle = \frac{1}{c} \sum_{\widetilde{\mathsf{v}}_0 \in B} (-1)^{\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle} = \frac{1}{2} - \frac{1}{2} = 0$$

We've shown that for any $\mathsf{v}_0, \mathsf{v}'_0 \in \mathbb{F}_2^n$, $\langle \mathsf{v}'_0 | U^\dagger U |\mathsf{v}_0\rangle = \mathbb{1}_{\mathsf{v}_0 == \mathsf{v}'_0}$, so U is unitary. □

Lemma 5.11. *For any $\mathsf{v}_0, \mathsf{v}'_0 \in A$, if $\mathsf{v}_0 \neq \mathsf{v}'_0$, then for exactly half of the vectors $\widetilde{\mathsf{v}}_0 \in B$, $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle = 1$, and for the other half, $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle = 0$.*

Proof. First, $\mathsf{v}_0 - \mathsf{v}'_0 \in A$ because A is a subspace. Second, for any non-zero $\mathsf{v}_0 - \mathsf{v}'_0 \in A$, there exists a $\widetilde{\mathsf{v}}^*_0 \in B$ for which $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}^*_0 \rangle = 1$. Otherwise, $\mathsf{v}_0 - \mathsf{v}'_0 \in B^\perp$, which is ruled out by Lemma 5.12.

Since B is a subspace, B can be partitioned into pairs of vectors $(\widetilde{\mathsf{v}}_0, \widetilde{\mathsf{v}}'_0)$ where $\widetilde{\mathsf{v}}'_0 = \widetilde{\mathsf{v}}_0 + \widetilde{\mathsf{v}}^*_0$. Next,

$$\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}'_0 \rangle \neq \langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle$$

Therefore, half of the vectors $\widetilde{\mathsf{v}}_0 \in B$ satisfy $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle = 1$, and the other half satisfy $\langle \mathsf{v}_0 - \mathsf{v}'_0, \widetilde{\mathsf{v}}_0 \rangle = 0$. □

Lemma 5.12. $A \cap B^\perp = \{\mathbf{0}\}$

Proof. First, $S^\perp \cap \mathsf{co}(S^\perp) = \{\mathbf{0}\}$ because $S^\perp$ shares exactly one vector with $\mathsf{co}(S^\perp)$ (Lemma 5.1). It has to be $\mathbf{0}$ because $S^\perp$ and $\mathsf{co}(S^\perp)$ are subspaces. By the same logic, $T^\perp \cap \mathsf{co}(T^\perp) = \{\mathbf{0}\}$.

Second,

$$B^\perp = [\mathsf{co}(T^\perp) \cap S^\perp]^\perp = [\mathsf{co}(T)^\perp \cap S^\perp]^\perp = \mathsf{span}[\mathsf{co}(T), S]$$

Next, let $\mathsf{z} \in A \cap B^\perp$. That means z can be decomposed into two vectors $(\mathsf{z}', \mathsf{z}'') \in \mathsf{co}(T) \times S$ such that $\mathsf{z} = \mathsf{z}' + \mathsf{z}''$. Additionally, z is in T and $\mathsf{co}(S)$.

We'll use these properties to show that $\mathsf{z} = \mathbf{0}$. $\mathsf{z}'' \in S \leq T$, so $\mathsf{z}' = \mathsf{z} - \mathsf{z}'' \in T$. But $\mathsf{z}' \in \mathsf{co}(T)$ as well, so $\mathsf{z}' = \mathbf{0}$. Therefore, $\mathsf{z} = \mathsf{z}'' \in S$. But $\mathsf{z} \in \mathsf{co}(S)$ as well, so $\mathsf{z} = \mathbf{0}$. Therefore, $A \cap B^\perp = \{\mathbf{0}\}$. □

Lemma 5.13. *Applying U to the v_0 register of $|\psi\rangle_0$ produces $|\psi\rangle_1$:*

$$\frac{1}{\sqrt{c}} \sum_{\mathsf{v}_0 \in A} |S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle \otimes (U|\mathsf{v}_0\rangle) = \frac{1}{\sqrt{c}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle |\widetilde{\mathsf{v}}_0\rangle$$

Proof.

$$\text{LHS} = \frac{1}{\sqrt{c \cdot |S|}} \sum_{\mathsf{v}_0 \in A} \sum_{\mathsf{z} \in S+\mathsf{v}_0+\mathsf{u}} |\mathsf{z}\rangle \otimes (U|\mathsf{v}_0\rangle) \cdot (-1)^{\langle \mathsf{z},\mathsf{w}\rangle} \tag{1}$$

$$= \frac{1}{\sqrt{c \cdot |S|}} \sum_{\mathsf{z} \in T+\mathsf{u}} \sum_{\mathsf{v}_0 \in A \,:\, \mathsf{z} \in S+\mathsf{v}_0+\mathsf{u}} |\mathsf{z}\rangle \otimes (U|\mathsf{v}_0\rangle) \cdot (-1)^{\langle \mathsf{z},\mathsf{w}\rangle} \tag{2}$$

$$= \frac{1}{\sqrt{c \cdot |S|}} \sum_{\mathsf{z} \in T+\mathsf{u}} |\mathsf{z}\rangle \otimes (U|\mathsf{v}_0\rangle) \cdot (-1)^{\langle \mathsf{z},\mathsf{w}\rangle} \quad \text{where } \mathsf{v}_0 \in A \text{ satisfies } \mathsf{z} \in S + \mathsf{v}_0 + \mathsf{u} \tag{3}$$

For Eq. (2) we switched the order of summation. z only takes values in $T + \mathsf{u}$, and it takes each value in $T + \mathsf{u}$ exactly once. For Eq. (3), we removed the summation over v_0 because it is uniquely determined by z. These facts are proven in Lemma 5.5.

Next:

$$\text{LHS} = \frac{1}{\sqrt{c^2 \cdot |S|}} \sum_{\widetilde{\mathsf{v}}_0 \in B} \sum_{\mathsf{z} \in T+\mathsf{u}} |\mathsf{z}\rangle |\widetilde{\mathsf{v}}_0\rangle \cdot (-1)^{\langle \mathsf{z},\mathsf{w}\rangle + \langle \mathsf{v}_0,\widetilde{\mathsf{v}}_0\rangle} \tag{4}$$

$$= \frac{1}{\sqrt{c^2 \cdot |S|}} \sum_{\widetilde{\mathsf{v}}_0 \in B} \sum_{\mathsf{z} \in T+\mathsf{u}} |\mathsf{z}\rangle |\widetilde{\mathsf{v}}_0\rangle \cdot (-1)^{\langle \mathsf{z},\widetilde{\mathsf{v}}_0+\mathsf{w}\rangle} \tag{5}$$

$$= \sqrt{\frac{|T|}{c^2 \cdot |S|}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle |\widetilde{\mathsf{v}}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle |\widetilde{\mathsf{v}}_0\rangle \tag{6}$$

$$= \text{RHS} \tag{7}$$

For Eq. (5), we used the fact that $\langle \mathsf{z}, \widetilde{\mathsf{v}}_0\rangle = \langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0\rangle$ (Lemma 5.14). □

Lemma 5.14. *Let $\mathsf{v}_0 \in A$, $\widetilde{\mathsf{v}}_0 \in B$, and $\mathsf{z} \in S + \mathsf{v}_0 + \mathsf{u}$. Then $\langle \mathsf{z}, \widetilde{\mathsf{v}}_0\rangle = \langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0\rangle$.*

Proof. First, $\mathsf{z} - \mathsf{v}_0 \in S + \mathsf{u} \subset \mathsf{span}[S, \mathsf{co}(T)]$. Second,

$$\mathsf{span}[S, \mathsf{co}(T)] = [S^\perp \cap \mathsf{co}(T)^\perp]^\perp = [S^\perp \cap \mathsf{co}(T^\perp)]^\perp = B^\perp$$

Therefore, $\langle \mathsf{z} - \mathsf{v}_0, \widetilde{\mathsf{v}}_0\rangle = 0$, so $\langle \mathsf{z}, \widetilde{\mathsf{v}}_0\rangle = \langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0\rangle$. □

6 General Compiler for Certified Deletion

In this section, we present a general technique for proving certified deletion that works well with existing cryptographic primitives and enables the constructions in subsequent sections.

Consider the following simple construction. To hide a bit b, we give the adversary:

$$|S_{\mathsf{v},\mathsf{w}}\rangle, b \oplus \langle \mathsf{v}, \mathbf{1} \rangle$$

where $|S_{\mathsf{v},\mathsf{w}}\rangle$ is a subspace coset state, sampled uniformly at random such that $\dim(S) = n/2$, $\mathsf{v} \in \mathsf{co}(S), \mathsf{w} \in \mathsf{co}(S^\perp)$. The bit b is masked by $\langle \mathsf{v}, \mathbf{1} \rangle$, and the information needed to remove the mask is stored in the subspace coset state.

To prove deletion, the adversary measures the subspace coset state in the Hadamard basis to get a vector $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$. We'll show that if they prove deletion, then all but negligible information about v is lost. That is, even if S is leaked at a later point, b remains statistically hidden because $\langle \mathsf{v}, \mathbf{1} \rangle$ is statistically close to uniformly random.

Definition 6.1 below describes this scenario. We say that security holds if the output of EXP^* is statistically close between the cases where $b = 0$ and $b = 1$.

Definition 6.1 (Certified Deletion Game, Abstract Form)**.** *Let b be a bit, let $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ be a QPT adversary, and let $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ be a quantum or classical operation. Then let $\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, b)$ be the output of the following experiment:*

1. *Challenge: Let $n = 4\lambda$. The challenger samples a subspace S of dimension $n/2$ along with vectors $(\mathsf{v}, \mathsf{w}) \leftarrow \mathsf{co}(S) \times \mathsf{co}(S^\perp)$, uniformly at random.*

   Next, they sample a subspace T uniformly at random such that $S \leq T$ and $\dim(T) = 3n/4$, using the procedure in Definition 5.2. Let $\mathsf{u} \in \mathsf{co}(T)$ be the unique coset such that $S + \mathsf{v} \subset T + \mathsf{u}$.

   Finally, the challenger sends the adversary the following challenge:

   $$|S_{\mathsf{v},\mathsf{w}}\rangle, \mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1} \rangle)$$

2. *Response: The adversary, running $\mathcal{A}_\lambda$, responds with a deletion certificate $\widetilde{\mathsf{z}} \in \mathbb{F}_2^n$ and an auxiliary state ρ.*

3. *Outcome: The challenger checks that*

   $$\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$$

   If so, they output $(\rho, S, T, \mathsf{u}, \mathsf{w}, b \oplus \langle \mathsf{v}, \mathbf{1} \rangle)$; if not, they output $\perp$.

In Definition 6.1, $\mathcal{Z}$ represents the side information given to the adversary. In the simplest case, $\mathcal{Z} = \perp$, and it's simple to prove that security holds. Note that we cannot give $S, \mathsf{v},$ or w in the clear because then the adversary could learn v while also outputting a $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$. In all of our applications, we need to give the adversary some information about $(S, \mathsf{v}, \mathsf{w})$, but we're careful to hide it. For instance, when $\mathcal{Z} = \mathsf{Enc}(1^\lambda, S)$ for some semantically-secure encryption scheme Enc, or $\mathcal{Z} = [\mathsf{Enc}(1^\lambda, S), i\mathcal{O}(P_{S^\perp + \mathsf{w}})]$, we can prove that security holds.

Theorem 6.2 (Semantically Secure Encryption)**.** *Let Enc be a semantically secure encryption scheme. That is to say: for all QPT $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$\left| \Pr\left[\mathcal{A}_\lambda(\mathsf{Enc}(1^\lambda, 0)) = 1\right] - \Pr\left[\mathcal{A}_\lambda(\mathsf{Enc}(1^\lambda, 1)) = 1\right] \right| = \mathrm{negl}(\lambda).$$

Next for any $\lambda \in \mathbb{N}$, let

$$\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b') = \mathsf{Enc}(1^\lambda, S).$$

Then for any QPT adversary $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,

$$\mathsf{TD}\left(\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)\right) = \mathrm{negl}(\lambda).$$

The next theorem says that we can make the deletion certificate publicly verifiable, assuming post-quantum $\mathsf{i}\mathcal{O}$. To do so, we include $\mathsf{i}\mathcal{O}(P_{S^\perp+\mathsf{w}})$ in $\mathcal{Z}$.

Theorem 6.3 (Encryption with Publicly-Verifiable Deletion)**.** *Let* Enc *be a semantically secure encryption scheme, and assume post-quantum indistinguishability obfuscation ($\mathsf{i}\mathcal{O}$). Next, for any $\lambda \in \mathbb{N}$, let*

$$\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b') = \left[\mathsf{Enc}(1^\lambda, S), \mathsf{i}\mathcal{O}(P_{S^\perp+\mathsf{w}})\right].$$

Then for any QPT adversary $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,

$$\mathsf{TD}\left(\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)\right) = \mathrm{negl}(\lambda).$$

It is natural to wonder whether we can include $\mathsf{i}\mathcal{O}(P_{S+\mathsf{v}})$ in $\mathcal{Z}$, just like we included $\mathsf{i}\mathcal{O}(P_{S^\perp+\mathsf{w}})$. In fact, this is not allowed because $\mathsf{i}\mathcal{O}$ only hides v computationally. If $\langle \mathsf{v}, \mathbf{1}\rangle$ is not statistically hidden, then neither is b. However, $\mathcal{Z}$ can include (T, u), which satisfy $S + \mathsf{v} \subset T + \mathsf{u}$, and for our applications that is good enough. The bit b remains statistically hidden because even given $(S, T, \mathsf{u}, \mathsf{w})$, $\langle \mathsf{v}, \mathbf{1}\rangle$ is uniformly random (with overwhelming probability over the choice of (S, T)).

6.1 General Theorem

The previous theorems are special cases of Theorem 6.5 below. We will use it in subsequent sections to prove certified deletion. Theorem 6.5 says that security holds in EXP^* if any information that $\mathcal{Z}$ gives the adversary about $S^\perp + \mathsf{w}$ could also be computed from a larger random coset $R + \mathsf{x}$ that contains $S^\perp + \mathsf{w}$. We call this property *subspace hiding*, and it is analogous to [Zha19]'s notion of subspace-hiding obfuscation (Definition 4.8). Below, we will precisely define the property of $\mathcal{Z}$ we need.

Definition 6.4 (Subspace Hiding)**.** *Let $\mathscr{A}$ be a class of adversaries*[13]*. We say that a quantum operation $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ is* ***subspace-hiding*** *for $\mathscr{A}$ if there exists a simulator $\{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ such that for any adversary in $\mathscr{A}$, their advantage in the following game is negligible in λ:*

1. *Let $n = 4\lambda$. The adversary chooses subspaces $S, T \leq \mathbb{F}_2^n$ such that $S \leq T, \dim(S) = n/2$, and $\dim(T) = 3n/4$, they choose vectors $\mathsf{u} \in \mathsf{co}(T)$ and $\mathsf{w} \in \mathsf{co}(S^\perp)$, and they choose a bit b'. Then they send these variables to the challenger.*

2. *The challenger samples R, a uniformly random superspace of $S^\perp$ of dimension $3n/4$. Let $\mathsf{x} \in \mathsf{co}(R)$ be the unique coset such that $S^\perp + \mathsf{w} \subset R + \mathsf{x}$. Next, the challenger samples a bit $c \leftarrow \{0, 1\}$. If $c = 0$, they compute $\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$ and send the output to the challenger. If $c = 1$, they compute $\mathcal{S}_\lambda(R, T, \mathsf{u}, \mathsf{x}, b')$ and send the output to the challenger.*

[13] $\mathscr{A}$ should be closed under constant-factor increases in space and time. That is say: for any adversary $\{\mathcal{A}\}_{\lambda\in\mathbb{N}} \in \mathscr{A}$ and any quantum adversary $\{\mathcal{B}\}_{\lambda\in\mathbb{N}}$, if the time and space complexity of $\{\mathcal{B}\}_{\lambda\in\mathbb{N}}$ is more than that of $\{\mathcal{A}\}_{\lambda\in\mathbb{N}}$ by a constant factor, then $\{\mathcal{B}\}_{\lambda\in\mathbb{N}}$ is also in $\mathscr{A}$.

3. *The adversary outputs a guess $c' \in \{0,1\}$ for c.*

The adversary's advantage is $|\Pr(c' = c) - 1/2|$.

Finally, the theorem below says that b is statistically hidden in EXP^* if $\mathcal{Z}$ is subspace-hiding.

Theorem 6.5 (General theorem). *Let $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ be defined as it was in EXP^*, and let $\mathscr{A}$ be a class of adversaries. Next, if $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ is subspace-hiding for $\mathscr{A}$, then for any adversary $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}} \in \mathscr{A}$,*

$$\mathsf{TD}\left(\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)\right) = \mathrm{negl}(\lambda)$$

6.2 Oracle version

So far, we have just defined the output of $\mathcal{Z}$ and $\mathcal{S}$ to be some quantum register that is sent to the adversary. We can consider a more general version where the output of $\mathcal{Z}$ and $\mathcal{S}$ also includes the description of a classical oracle $\mathcal{O}$, which the adversary gets (quantum-accessible) query access to.

It is simple to modify our definitions for the oracle case. In Definition 6.1 and Definition 6.4, we said that the challenger *sends* the output of $\mathcal{Z}$ or $\mathcal{S}$ to the adversary. But if the output includes the description of an oracle, then the challenger gives $\mathcal{A}$ quantum query access to the oracle rather than sending the description of the oracle to $\mathcal{A}$. Finally, Theorem 6.5 and its proof still hold when the output of $\mathcal{Z}$ and $\mathcal{S}$ includes an oracle.

6.3 Proof of Theorem 6.5

See Section 2.2 for an overview of how this proof works.

Delayed Preparation of Coset States

First, we will recall the following definitions from Section 5. Define $|\psi\rangle_0$ and $|\psi\rangle_1$ as follows:

$$|\psi\rangle_0 = \frac{1}{\sqrt{|A|}} \sum_{\mathsf{v}_0 \in A} |S_{\mathsf{u}+\mathsf{v}_0,\mathsf{w}}\rangle\, |\mathsf{v}_0\rangle$$

$$|\psi\rangle_1 = \frac{1}{\sqrt{|B|}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |T_{\mathsf{u},\widetilde{\mathsf{v}}_0+\mathsf{w}}\rangle\, |\widetilde{\mathsf{v}}_0\rangle$$

where, $A = \mathsf{co}(S) \cap T$ and $B = \mathsf{co}(T^\perp) \cap S^\perp$.

Next, we will define a unitary U that acts on the second register and maps superpositions over to A to superpositions over B. For any $\mathsf{v}_0 \in A$, let

$$U\,|\mathsf{v}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\widetilde{\mathsf{v}}_0 \in B} |\widetilde{\mathsf{v}}_0\rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0\rangle}$$

and for any $\widetilde{\mathsf{v}}_0 \in B$, let

$$U^\dagger\,|\widetilde{\mathsf{v}}_0\rangle = \frac{1}{\sqrt{c}} \sum_{\mathsf{v}_0 \in A} |\mathsf{v}_0\rangle \cdot (-1)^{\langle \mathsf{v}_0, \widetilde{\mathsf{v}}_0\rangle}$$

Notes: c is a normalization constant, and U depends on (S,T). Technically, U acts on any superposition over $\mathbb{F}_2^n$, and we define it fully in Definition 5.8. Also see Lemma 5.9 and Lemma 5.10.

Finally, when U is applied to the second register of $|\psi\rangle_0$, it maps the state to $|\psi\rangle_1$ (Lemma 5.13).

Hybrids

Consider the following sequence of hybrids. We'll only say how each hybrid differs from the one before it:

- $\mathcal{H}_0(b)$ is $\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, b)$.
- $\mathcal{H}_1(b)$: In the *challenge* stage, the challenger samples a bit $b' \leftarrow \{0,1\}$, and sends the adversary the following challenge:
$$|S_{\mathsf{v},\mathsf{w}}\rangle\,, \mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$$
Then at the start of the *outcome* stage, they check whether $b' = b \oplus \langle \mathsf{v}, \mathbf{1}\rangle$. If so, they proceed normally. If not, the game stops, and the outcome is $\perp$.
- $\mathcal{H}_2(b)$: In the *challenge* stage, the challenger samples $(S, T, \mathsf{u}, \mathsf{w}, b')$ normally, but then they jointly sample $(|S_{\mathsf{v},\mathsf{w}}\rangle\,, \mathsf{v})$ as follows. First they prepare $|\psi\rangle_0$ and then they measure the v_0 register. Let $\mathsf{v} = \mathsf{u} + \mathsf{v}_0$, and then the first register of $|\psi\rangle_0$ holds the state $|S_{\mathsf{v},\mathsf{w}}\rangle$. Then the challenge sent to the adversary is the same as in the previous hybrid.
- $\mathcal{H}_3(b)$: In the *challenge* stage, the challenger prepares $|\psi\rangle_0$ as before, but they delay measuring the v_0 register until the start of the *outcome* stage. The challenge still includes the $|S_{\mathsf{v},\mathsf{w}}\rangle$ register, but now it's entangled with the v_0 register.
- $\mathcal{H}_4$: At the start of the *outcome* stage, the challenger applies U to the v_0 register and measures it to get $\widetilde{\mathsf{v}}_0$. Also let $\widetilde{\mathsf{v}} = \widetilde{\mathsf{v}}_0 + \mathsf{w}$. Then they check that
$$\widetilde{\mathsf{z}} \in (S^\perp + \mathsf{w}) \backslash (T^\perp + \widetilde{\mathsf{v}})$$
The output of the hybrid is 1 if the check passes and 0 otherwise.
- $\mathcal{H}_5$: The challenger applies U to the v_0 register and measures the result in the *challenge* stage rather than the *outcome* stage. Therefore the challenge is:
$$|T_{\mathsf{u},\widetilde{\mathsf{v}}}\rangle\,, \mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$$
for a uniformly random $\widetilde{\mathsf{v}}_0 \leftarrow B$.
- $\mathcal{H}_6$: In the *challenge* stage, the challenger samples $(S, T, \mathsf{u}, \widetilde{\mathsf{v}}, \mathsf{w}, b')$ as before. They also sample a subspace R uniformly at random such that $S^\perp \leq R$, and $\dim(R) = 3n/4$, using the procedure in Definition 5.2. Let $\mathsf{x} \in \mathsf{co}(R)$ be the unique coset such that $S^\perp + \mathsf{w} \subset R + \mathsf{x}$. Then the challenge is
$$|T_{\mathsf{u},\widetilde{\mathsf{v}}}\rangle\,, \mathcal{S}_\lambda(R, T, \mathsf{u}, \mathsf{x}, b')$$

The following claims step through the hybrids above to show that

$$\mathsf{TD}\left(\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)\right) = \mathrm{negl}(\lambda)$$

Claim 6.6. $\Pr(\mathcal{H}_6 \to 1) = \mathrm{negl}(\lambda)$.

Proof. The adversary's challenge is completely determined by $(R, T, \mathsf{u}, \widetilde{\mathsf{v}}, \mathsf{x}, b')$. For the rest of the proof, fix any valid choice of these variables. Then given those variables, $\widetilde{\mathsf{z}}$ is independent of $S^\perp + \mathsf{w}$, and $S^\perp + \mathsf{w}$ is a uniformly random subspace coset such that $\dim(S^\perp) = n/2$, $T^\perp \leq S^\perp \leq R$, and $T^\perp + \widetilde{\mathsf{v}} \subset S^\perp + \mathsf{w} \subset R + \mathsf{x}$. Then we can show that the adversary has negligible probability of winning $\mathcal{H}_6$:

$$\begin{aligned}\Pr[\mathcal{H}_6 \to 1] &= \Pr_{S,\mathsf{w}}\left[\,\widetilde{\mathsf{z}} \in (S^\perp + \mathsf{w}) \backslash (T^\perp + \widetilde{\mathsf{v}})\,\right] \leq \max_{\mathsf{z} \in \mathbb{F}_2^n} \Pr_{S,\mathsf{w}}\left[\,\mathsf{z} \in (S^\perp + \mathsf{w}) \backslash (T^\perp + \widetilde{\mathsf{v}})\,\right] \\ &= \max_{\mathsf{z} \in (R+\mathsf{x}) \backslash (T^\perp + \widetilde{\mathsf{v}})} \Pr_{S,\mathsf{w}}\left(\mathsf{z} \in S^\perp + \mathsf{w}\right) = \frac{|S^\perp + \mathsf{w}| - |T^\perp + \widetilde{\mathsf{v}}|}{|R + \mathsf{x}| - |T^\perp + \widetilde{\mathsf{v}}|} = \frac{2^{n/2} - 2^{n/4}}{2^{3n/4} - 2^{n/4}} = \mathrm{negl}(\lambda)\end{aligned}$$

□

Claim 6.7. $\Pr(\mathcal{H}_5 \to 1) = \mathrm{negl}(\lambda)$.

Proof. If $\Pr[\mathcal{H}_5 \to 1]$ and $\Pr[\mathcal{H}_6 \to 1]$ were non-negligibly different, then that would break the subspace-hiding property of $\mathcal{Z}$. The reduction goes as follows: first, the adversary in the subspace-hiding game would sample $(S, T, \mathsf{u}, \widetilde{\mathsf{v}}, \mathsf{w}, b')$ from the same distribution as in $\mathcal{H}_5/\mathcal{H}_6$. Then they send $(S, T, \mathsf{u}, \mathsf{w}, b')$ to the subspace-hiding challenger to receive either $\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$ or $\mathcal{S}_\lambda(R, T, \mathsf{u}, \mathsf{x}, b')$. Then they simulate the rest of the hybrid, either $\mathcal{H}_5$ or $\mathcal{H}_6$. Finally their output c is the output of the hybrid. If for some adversary, $\Pr[\mathcal{H}_5 \to 1]$ and $\Pr[\mathcal{H}_6 \to 1]$ were non-negligibly different, then this reduction would break the subspace-hiding property of $\mathcal{Z}_\lambda$. □

Claim 6.8. $\Pr(\mathcal{H}_4 \to 1) = \Pr(\mathcal{H}_5 \to 1)$

Proof. The only difference between the hybrids is which comes first: (1) the adversary's computation, or (2) the act of applying U to the v_0 register and measuring it. These operations commute because they act on separate registers, so the outputs of these hybrids are identically distributed. □

Claim 6.9. $\mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)] = \mathrm{negl}(\lambda)$

Proof.

1. Let us run $\mathcal{H}_3$ and $\mathcal{H}_4$ using the same QPT adversary. Then the hybrids are identical until the start of the *outcome* stage. It will be useful to compare the outcome stages of the two hybrids.
2. $\mathcal{H}_4$ outputs 0 with overwhelming probability, so we will focus on this event. If $\mathcal{H}_4$ outputs 0, then one of the following events occurred:
   (a) Event a: $\widetilde{\mathsf{z}} \notin S^\perp + \mathsf{w}$
   (b) Event b: $\widetilde{\mathsf{z}} \in T^\perp + \widetilde{\mathsf{v}}_0 + \mathsf{w}$

   These two events are mutually exclusive because $T^\perp + \widetilde{\mathsf{v}}_0 + \mathsf{w} \subseteq S^\perp + \mathsf{w}$ for any $\widetilde{\mathsf{v}}_0 \in B$.

   Next, if event b occurred, then the value of $\widetilde{\mathsf{v}}_0$ is uniquely determined by $(S, T, \mathsf{w}, \widetilde{\mathsf{z}})$ (Lemma 5.6). For a given $(S, T, \mathsf{w}, \widetilde{\mathsf{z}})$ satisfying $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$, let $\widetilde{\mathsf{v}}_0^*$ be the unique value of $\widetilde{\mathsf{v}}_0 \in B$ for which $\widetilde{\mathsf{z}} \in T^\perp + \widetilde{\mathsf{v}}_0 + \mathsf{w}$.

3. We can view the *challenge* and *response* stages of $\mathcal{H}_3/\mathcal{H}_4$ as a procedure for generating a quantum state, which includes all of the registers held by the adversary and challenger, including the registers holding classical random variables. Next, we can view the *outcome* stage of $\mathcal{H}_4$ as a measurement performed on that state.

   We will define projections on the state that represent events a and b.

   (a) Let Π_a project onto values of $(S, T, \mathsf{w}, \widetilde{\mathsf{z}})$ for which $\widetilde{\mathsf{z}} \notin S^\perp + \mathsf{w}$.

   (b) Let Π_b:

      i. first project onto all values of $(S, T, \mathsf{w}, \widetilde{\mathsf{z}})$ for which $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$, and

      ii. then project the v_0 register onto $U \left|\widetilde{\mathsf{v}}_0^*\right\rangle \left\langle\widetilde{\mathsf{v}}_0^*\right| U^\dagger$.

   Note that $\widetilde{\mathsf{v}}_0^*$ is determined by the values on the $(S, T, \mathsf{w}, \widetilde{\mathsf{z}})$ registers.

   Note that Π_a and Π_b project onto orthogonal spaces because events a and b are mutually exclusive.

4. Let us assume for a moment that the state produced by the *challenge* and *response* stages lies in the image of Π_a. In this case, $\mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)] = 0$ because the output of $\mathcal{H}_3$ is $\perp$, which doesn't depend on b.

5. Next, consider the case where the state produced by the *challenge* and *response* stages lies in the image of Π_b. We will show that in this case, $\mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)] = \mathrm{negl}(\lambda)$.

   First, the state on the v_0 register is initially $U \left|\widetilde{\mathsf{v}}_0^*\right\rangle \left\langle\widetilde{\mathsf{v}}_0^*\right| U^\dagger$. Then the $\mathcal{H}_3$ challenger measures the v_0 register in the computational basis, which returns a uniformly random value in A that is independent of the other registers.[14] Furthermore, $\langle \mathsf{v}, \mathbf{1} \rangle$ is uniformly random, with overwhelming probability over the choice of (S, T) (Lemma 6.10).

   The only part of $\mathcal{H}_3$ that depends on b is when the challenger checks that $b' = b \oplus \langle \mathsf{v}, \mathbf{1} \rangle$. But if $\langle \mathsf{v}, \mathbf{1} \rangle$ is uniformly random and independent of the other registers, then the output of the hybrid is independent of b. Therefore, $\mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)] = \mathrm{negl}(\lambda)$.

6. Now consider the case where the state produced by the *challenge* and *response* stages is in the image of $\Pi_a + \Pi_b$. In this case as well, $\mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)] = \mathrm{negl}(\lambda)$. The output of $\mathcal{H}_3$ is obtained by tracing out all registers except for the output register. The result is a mixture of the output states from Item 4 and Item 5.

7. In reality, the state produced by the *challenge* and *response* stages is negligibly close in trace distance to a state in the image of $\Pi_a + \Pi_b$. This is because $\mathcal{H}_4 \to 0$ with overwhelming probability. Therefore $\Pi_a + \Pi_b$ is a gentle measurement: applying $\Pi_a + \Pi_b$ to the state produced by the *challenge* and *response* stages changes the state by a negligible amount, measured in trace distance.

   Therefore, $\mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)] = \mathrm{negl}(\lambda)$.

[14] Compared to prior work ([BI20, BK23])), our claim that v_0 is uniformly random and independent of the other registers is quite strong. Our use of subspace coset states allows us to make such a strong claim and makes our proof more streamlined. [BI20, BK23] essentially argue that the v_0 register is somewhat localized in the Hadamard basis, so the value in the computational basis has a certain amount of entropy, even conditioned on the adversary's registers; in contrast, we say that the v_0 register is a single eigenstate of the U basis, $U \left|\widetilde{\mathsf{v}}_0^*\right\rangle$, and therefore the value in the computational basis is uniformly random.

□

Lemma 6.10. *With overwhelming probability in n over the randomness of S and T, exactly half of the vectors $\mathsf{v} \in A + \mathsf{u}$ satisfy $\langle \mathsf{v}, \mathbf{1} \rangle = 1$, and the other half satisfy $\langle \mathsf{v}, \mathbf{1} \rangle = 0$.*

Proof. With overwhelming probability, A contains at least one vector v_0^* for which $\langle \mathsf{v}_0^*, \mathbf{1} \rangle = 1$. This is because over the randomness of S and T, A is a uniformly random subspace of dimension $n/4$. One way to sample such a subspace is to sample $n/4$ vectors uniformly at random such that they are linearly independent. Half of the vectors $\mathsf{z} \in \mathbb{F}_2^n$ satisfy $\langle \mathsf{z}, \mathbf{1} \rangle = 1$, so the probability that all the vectors in A satisfy $\langle \mathsf{v}_0, \mathbf{1} \rangle = 0$ is $\leq (1/2)^{n/4} = \mathrm{negl}(n)$.

If there is at least one $\mathsf{v}_0^* \in A$ for which $\langle \mathsf{v}_0^*, \mathbf{1} \rangle = 1$, then exactly half of the vectors $\mathsf{v}_0 \in A$ satisfy $\langle \mathsf{v}_0, \mathbf{1} \rangle = 1$. Since A is a subspace, it can be partitioned into pairs of vectors $(\mathsf{v}_0, \mathsf{v}_0')$ where $\mathsf{v}_0' = \mathsf{v}_0 + \mathsf{v}_0^*$. Next, $\langle \mathsf{v}_0, \mathbf{1} \rangle \neq \langle \mathsf{v}_0', \mathbf{1} \rangle$, and

$$\langle \mathsf{v}_0 + \mathsf{u}, \mathbf{1} \rangle \neq \langle \mathsf{v}_0' + \mathsf{u}, \mathbf{1} \rangle$$

Therefore, half of the vectors $\mathsf{v} \in A + \mathsf{u}$ satisfy $\langle \mathsf{v}, \mathbf{1} \rangle = 1$, and the other half satisfy $\langle \mathsf{v}, \mathbf{1} \rangle = 0$. □

Claim 6.11. $\frac{1}{2} \cdot \mathsf{TD}[\mathcal{H}_0(0), \mathcal{H}_0(1)] = \mathsf{TD}[\mathcal{H}_1(0), \mathcal{H}_1(1)] = \mathsf{TD}[\mathcal{H}_2(0), \mathcal{H}_2(1)] = \mathsf{TD}[\mathcal{H}_3(0), \mathcal{H}_3(1)]$

Proof. First, the only difference between $\mathcal{H}_3$ and $\mathcal{H}_2$ is which comes first: (1) the adversary's computation or (2) the act of measuring v_0. These operations commute because they act on separate registers, so the outputs of the two hybrids are identically distributed.

Second, the outputs of $\mathcal{H}_2$ and $\mathcal{H}_1$ are identically distributed. The only difference between the hybrids is how the challenge is sampled, but both hybrids ultimately sample the challenge from the same distribution.

Finally, $\frac{1}{2} \cdot \mathsf{TD}[\mathcal{H}_0(0), \mathcal{H}_0(1)] = \mathsf{TD}[\mathcal{H}_1(0), \mathcal{H}_1(1)]$. In $\mathcal{H}_1(b)$, when $b' = b \oplus \langle \mathsf{v}, \mathbf{1} \rangle$, the challenge and the output of the hybrid have the same distribution as in $\mathcal{H}_0(b)$. When $b' \neq b \oplus \langle \mathsf{v}, \mathbf{1} \rangle$, then the output is $\perp$. b' is independent of b and v, so

$$\mathsf{TD}[\mathcal{H}_1(0), \mathcal{H}_1(1)] = \frac{1}{2} \cdot \mathsf{TD}[\mathcal{H}_0(0), \mathcal{H}_0(1)] + \frac{1}{2} \cdot 0 = \frac{1}{2} \cdot \mathsf{TD}[\mathcal{H}_0(0), \mathcal{H}_0(1)]$$

□

In summary, we've shown that

$$\mathsf{TD}[\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)] = \mathsf{TD}[\mathcal{H}_0(0), \mathcal{H}_0(1)] = \mathrm{negl}(\lambda)$$

6.4 Proofs of Theorem 6.2 and Theorem 6.3

Theorem 6.12 (Theorem 6.2 Restated)**.** *Let* Enc *be a semantically secure encryption scheme. Next for any $\lambda \in \mathbb{N}$, let*

$$\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b') = \mathsf{Enc}(1^\lambda, S).$$

Then for any QPT adversary $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$,

$$\mathsf{TD}\left(\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)\right) = \mathrm{negl}(\lambda).$$

Proof. It suffices to prove that $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ is subspace-hiding for all QPT adversaries, and then we can appeal to Theorem 6.5 to finish the proof. For any $\lambda \in \mathbb{N}$, let the simulator be $S_\lambda(R, T, \mathsf{u}, \mathsf{x}, b') = \mathsf{Enc}(1^\lambda, 0^*)$, where 0^* is the same length as the description of S. Any QPT adversary has negligible advantage at distinguishing $\mathsf{Enc}(1^\lambda, S)$ and $\mathsf{Enc}(1^\lambda, 0^*)$, so $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ is subspace-hiding for all QPT adversaries. □

Theorem 6.13 (Theorem 6.3 Restated)**.** *Let* Enc *be a semantically secure encryption scheme, and assume post-quantum indistinguishability obfuscation (*i$\mathcal{O}$*). Next, for any $\lambda \in \mathbb{N}$, let*

$$\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b') = \left[\mathsf{Enc}(1^\lambda, S), \mathsf{i}\mathcal{O}(P_{S^\perp+\mathsf{w}})\right].$$

Then for any QPT adversary $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,

$$\mathsf{TD}\left(\mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 0), \mathsf{EXP}^*(1^\lambda, \mathcal{A}_\lambda, \mathcal{Z}_\lambda, 1)\right) = \mathrm{negl}(\lambda)$$

Proof. We only need to show that $\{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ is subspace-hiding for all QPT adversaries. Let the simulator be $S_\lambda(R, T, \mathsf{u}, \mathsf{x}, b') = [\mathsf{Enc}(1^\lambda, 0^*), \mathsf{i}\mathcal{O}(P_{R+\mathsf{x}})]$. We will show that no QPT adversary can distinguish the outputs of $\mathcal{Z}$ and $\mathcal{S}$ with non-negligible advantage. Consider the following sequence of hybrids:

- $\mathcal{H}_0$: Given $(S, T, \mathsf{u}, \mathsf{w}, b')$, compute and output $[\mathsf{Enc}(1^\lambda, S), \mathsf{i}\mathcal{O}(P_{S^\perp+\mathsf{w}})]$.
- $\mathcal{H}_1$: Given $(S, T, \mathsf{u}, \mathsf{w}, b')$, compute and output $[\mathsf{Enc}(1^\lambda, 0^*), \mathsf{i}\mathcal{O}(P_{S^\perp+\mathsf{w}})]$.
- $\mathcal{H}_2$: Given $(S, T, \mathsf{u}, \mathsf{w}, b')$, sample R, a uniformly random superspace of $S^\perp$ of dimension $3n/4$, and let $\mathsf{x} \in \mathsf{co}(R)$ be the unique coset for which $S^\perp + \mathsf{w} \subset R + \mathsf{x}$. Then compute and output $[\mathsf{Enc}(1^\lambda, 0^*), \mathsf{i}\mathcal{O}(P_{R+\mathsf{x}})]$.

Every QPT adversary has negligible advantage at distinguishing the outputs of $\mathcal{H}_0$ and $\mathcal{H}_1$, by the semantic security of Enc. Furthermore, every QPT adversary has negligible advantage at distinguishing the outputs of $\mathcal{H}_1$ and $\mathcal{H}_2$, by Corollary 4.10.. Essentially, this follows from the fact that i$\mathcal{O}$ is a subspace-hiding obfuscator

Next, $\mathcal{H}_0$ and $\mathcal{H}_2$ are the two cases the adversary is asked to distinguish in the subspace-hiding game (Definition 6.4), and we've shown that the advantage for any QPT adversary is negligible. Therefore, $\mathcal{Z}$ is subspace-hiding for any QPT adversary. □

Combined with constructions from [BK23], we obtain the following immediate corollary.

Corollary 6.14. *Assuming post-quantum indistinguishability obfuscation and $X \in \{$public-key, attribute-based, fully-homomorphic, timed-release, witness$\}$ encryption, there exists X with publicly-verifiable certifed deletion.*

7 Blind Delegation with Certified Deletion

Blind delegation is an interactive protocol that allows a client with limited computational resources to compute a resource-intensive function with the help of a more-powerful server. Through this protocol, the client learns the output of the function, but their running time is much less than the time needed to actually compute the function. To reduce the client's computational load, the

client delegates the most expensive computations to the server. However the server is blind, in the sense that they do not learn anything about the input to the function that they are computing.

There exist protocols for blind delegation based on fully homomorphic encryption, in which the server receives an encryption of the input. But these protocols only hide the input from the server computationally. Our version of blind delegation provides statistical hiding after deletion. After learning the output of the computation, the client can subsequently request that the server delete the encryption of the input and prove that they have done so. If the proof is accepted, it guarantees that the input is statistically hidden from the server, even if the client's secret key is leaked later on.

The rest of the section is organized as follows: we will first define maliciously-secure blind delegation with certified deletion, and then provide a construction and security proof.

7.1 Definitions

In this section, we will define the cryptographic primitive *maliciously secure blind delegation with certified deletion*. We will use a simulation-based definition, which is common in the study of interactive computation, and which encompasses both correctness and security.

The client wishes to learn $y = \mathcal{M}^t(x)$, the result of running Turing machine $\mathcal{M}$ on input x for t timesteps. The client's running time should be $\mathrm{poly}(\log t)$, which is much less than the time needed to actually compute y.

Ideal functionality. We define the desired input-output behavior of our protocol using the ideal functionality $\mathcal{F}_{\mathsf{BD}}$, which is given in Section 7.1. An ideal functionality is a classical interactive machine. It may occur over multiple phases, each with distinct inputs and outputs, and the outputs of previous phases may be used as inputs in later phases . The first phase of $\mathcal{F}_{\mathsf{BD}}$ is Setup, where the client encrypts x. The next is Eval, where the client asks the server to compute $\mathcal{M}^t(x)$. Eval may be repeated many times if the client wants to compute many Turing machines on x. Later on, we will add another phase to handle deletion.

Ideal Functionality $\mathcal{F}_{\mathsf{BD}}$

Parties: client C and server S, each with input the security parameter 1^λ.

- Setup: $\mathcal{F}_{\mathsf{BD}}$ receives an input x from C, and sends $|x|$ to S.
- Eval: Repeat the following an arbitrary number of times.
  - $\mathcal{F}_{\mathsf{BD}}$ receives a description of a Turing machine $\mathcal{M}$ and a step-size t from C, and sends $(\mathcal{M}, t)$ to S.[a]
  - $\mathcal{F}_{\mathsf{BD}}$ sends $y := \mathcal{M}^t(x)$ to C.

[a] For convenience, we keep $\mathcal{M}$ and t public, and only hide the client's input x. However, note that $\mathcal{M}$ could be the universal Turing machine, in which case hiding x would mean hiding everything about the client's computation except for an upper bound t on its run-time.

Figure 1: The ideal functionality for blind delegation with certified deletion.

Security with abort. If an adversarial server decides not to compute $\mathcal{M}^t(x)$, we want the client's output to be abort, so we add the following feature to $\mathcal{F}_{\mathsf{BD}}$: the ideal functionality knows when the server is corrupted. Then at the end of each phase, once it has computed the outputs, the functionality sends the outputs of the server to the adversary. Then the functionality awaits a command from the adversary of either deliver or abort. Upon receiving deliver, the functionality delivers to the client their outputs. Upon receiving abort, the functionality instead delivers abort to the client.

The real-ideal paradigm. A two-party protocol $\Pi_{\mathcal{F}}$ for computing the (potentially reactive) functionality $\mathcal{F}$ consists of two families of quantum interactive machines A and B. An adversary intending to attack the protocol by corrupting a party $M \in \{A, B\}$ can be described by a family of sequences of quantum interactive machines $\{\mathcal{A}_\lambda := (\mathcal{A}_{\lambda,1}, \ldots, \mathcal{A}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$, where k is the number of phases of $\mathcal{F}$. This adversarial interaction happens in the presence of an *environment*, which is a family of sequences of quantum operations $\{\mathcal{Z}_\lambda := (\mathcal{Z}_{\lambda,1}, \ldots, \mathcal{Z}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$, and a family of initial advice states $\{|\psi_\lambda\rangle\}_{\lambda\in\mathbb{N}}$. It proceeds as follows.

- $\mathcal{Z}_{\lambda,1}$ receives as input $|\psi_\lambda\rangle$. It outputs what (if any) inputs the honest party $H \in \{A, B\}$ is initialized with for the first phase of $\Pi_{\mathcal{F}}$. It also outputs a quantum state on registers (A, Z), where A is the state that holds inputs and outputs of the adversary, and Z holds the remaining state of the environment.
- $\mathcal{A}_{\lambda,1}$ receives as input a state on register A, and interacts with the honest party in the first phase of $\Pi_{\mathcal{F}}$. It may also maintain an additional register A'. It outputs a state on register A.
- $\mathcal{Z}_{\lambda,2}$ receives as input registers (A, Z) along with the honest party outputs from the first phase. It computes honest party inputs for the second phase, and updates registers (A, Z).
- $\mathcal{A}_{\lambda,2}, \mathcal{Z}_{\lambda,3}, \ldots, \mathcal{A}_{\lambda,k}$ are defined analogously.

Given an adversary, environment, and advice, we define the random variable $\Pi_{\mathcal{F}}[\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ as the output of the above procedure, which includes registers (A, Z) and the final honest party outputs.

An *ideal-world* protocol $\widetilde{\Pi}_{\mathcal{F}}$ for functionality $\mathcal{F}$ consists of "dummy" parties $\widetilde{A}$ and $\widetilde{B}$ that have access to an additional "trusted" party that implements $\mathcal{F}$. That is, $\widetilde{A}$ and $\widetilde{B}$ only interact directly with $\mathcal{F}$, providing inputs and receiving outputs, and do not interact with each other. We consider the execution of ideal-world protocols in the presence of a simulator, described by a family of sequences of quantum interactive machines $\{\mathcal{S}_\lambda := (\mathcal{S}_{\lambda,1}, \ldots, \mathcal{S}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$, analogous to the definition of an adversary above. This interaction also happens in the presence of an environment $\{\mathcal{Z}_\lambda := (\mathcal{Z}_{\lambda,1}, \ldots, \mathcal{Z}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$, and a family of initial advice states $\{|\psi_\lambda\rangle\}_{\lambda\in\mathbb{N}}$, as described above. Note that the simulator may maintain a register between phases that the environment cannot access. Finally, we define the analogous random variable $\widetilde{\Pi}_{\mathcal{F}}[\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ to be the output of the ideal-world procedure.

Definition 7.1 (Secure realization of $\mathcal{F}$)**.** *A protocol $\Pi_{\mathcal{F}}$ securely realizes the k-phase functionality $\mathcal{F}$ if the following property holds:*

- ***Computational security.*** *For every QPT adversary* $\{\mathcal{A}_\lambda := (\mathcal{A}_{\lambda,1},\dots,\mathcal{A}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$ *corrupting either party* A *or* B*, there exists a QPT simulator* $\{\mathcal{S}_\lambda := (\mathcal{S}_{\lambda,1},\dots,\mathcal{S}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$ *such that for any QPT environment* $\{\mathcal{Z}_\lambda := (\mathcal{Z}_{\lambda,1},\dots,\mathcal{Z}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$*, polynomial-size family of advice* $\{|\psi_\lambda\rangle\}_{\lambda\in\mathbb{N}}$*, and QPT distinguisher* $\{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$*, it holds that*

$$\left|\Pr\left[\mathcal{D}_\lambda\left(\Pi_{\mathcal{F}}[\mathcal{A}_\lambda,\mathcal{Z}_\lambda,|\psi_\lambda\rangle]\right)=1\right]-\Pr\left[\mathcal{D}_\lambda\left(\widetilde{\Pi}_{\mathcal{F}}[\mathcal{S}_\lambda,\mathcal{Z}_\lambda,|\psi_\lambda\rangle]\right)=1\right]\right|=\mathrm{negl}(\lambda).$$

The deletion phase. We will describe how to add a deletion phase to a *generic* two-party functionality, although the reader may keep in mind the specific functionality, for blind delegation, that we defined above. We use $\mathcal{F}^{\mathsf{Del}}$ to denote a functionality $\mathcal{F}$ with an added deletion phase.

Section 7.1 describes the deletion phase. When the parties are labeled A and B, we use the convention that A requests deletion, and then B deletes A's information. If $\mathcal{F}$ is reactive, we allow A to request the deletion phase between any two phases of $\mathcal{F}$, or at the end of $\mathcal{F}$. But once the deletion phase has been executed, the functionality ends, and no other phases will be executed.

Deletion Phase

Parties: A and B

- Receive a query Deletion_Requested from A, and send it to B.
- Receive a query Deletion_Confirmed from B. If a message Deletion_Requested has been recorded, send Deletion_Confirmed to A, and otherwise ignore the message.

Figure 2: Specification of a generic deletion phase that can be added to any ideal functionality $\mathcal{F}$.

The deletion phase adds one bit to each party's output. Party B's output is denoted DelReq, which is set to 1 if party A sends a message Deletion_Requested (and is set to 0 otherwise). Party A's output is denoted DelOutcome, and is set to 1 if party B sends a message Deletion_Confirmed (and is set to 0 otherwise).

The long-term secrets tape. In certified deletion, we want security to hold even when the honest party's secret information is leaked, so we will define $\mathbf{sec}$ to represent the leaked information. Let $\mathbf{sec}$ be a (classical or quantum) tape where the honest party writes all of its "long-term secrets". That is, at the end of each message, the honest party writes all the information it needs to participate in the rest of the protocol on $\mathbf{sec}$, and at the beginning of its next message computation, it retrieves all information it needs from this tape.

Definition 7.2 (Secure realization of $\mathcal{F}^{\mathsf{Del}}$)**.** *A protocol* $\Pi_{\mathcal{F}^{\mathsf{Del}}}$ securely realizes *the* k*-phase functionality* $\mathcal{F}^{\mathsf{Del}}$ *if the following properties hold:*

- ***Computational security.*** $\mathcal{F}^{\mathsf{Del}}$ *satisfies the computational security property of Definition 7.1.*
- ***Certified deletion.*** *For every QPT adversary* $\{\mathcal{A}_\lambda := (\mathcal{A}_{\lambda,1},\dots,\mathcal{A}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$ *corrupting party* B*, there exists a QPT simulator* $\{\mathcal{S}_\lambda := (\mathcal{S}_{\lambda,1},\dots,\mathcal{S}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$ *such that for any QPT environment* $\{\mathcal{Z}_\lambda := (\mathcal{Z}_{\lambda,1},\dots,\mathcal{Z}_{\lambda,k})\}_{\lambda\in\mathbb{N}}$*, and polynomial-size family of advice* $\{|\psi_\lambda\rangle\}_{\lambda\in\mathbb{N}}$*,*

$$\mathsf{TD}\left(\Pi^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}}[\mathcal{A}_\lambda,\mathcal{Z}_\lambda,|\psi_\lambda\rangle],\widetilde{\Pi}^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}}[\mathcal{S}_\lambda,\mathcal{Z}_\lambda,|\psi_\lambda\rangle]\right)=\mathrm{negl}(\lambda),$$

where

- $\Pi^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}}[\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ *is defined to consist of* $\Pi_{\mathcal{F}^{\mathsf{Del}}}[\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ ***as well as the contents of party*** A***'s tape*** $\mathbf{sec}$ *if party* A*'s output* $\mathsf{DelOutcome}$ *is set to 1, and is defined to be* $\bot$ *otherwise, and*
- $\widetilde{\Pi}^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}}[\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ *is defined to be equal to* $\widetilde{\Pi}_{\mathcal{F}^{\mathsf{Del}}}[\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ *if party* A*'s output* $\mathsf{DelOutcome}$ *is set to 1 (that is, the simulator must simulate* $\mathbf{sec}$*), and is defined to be* $\bot$ *otherwise.*

Finally, we define what it means for an interactive protocol Π between client and server to be a *maliciously secure blind delegation protocol with certified deletion*.

Definition 7.3. *A two-party protocol* Π *between a client and a server is a secure protocol for blind delegation with certified deletion if the following properties hold.*

- ***Security.*** Π *securely realizes (Definition 7.2) the functionality* $\mathcal{F}^{\mathsf{Del}}_{\mathsf{BD}}$ *against adversaries that corrupt the server (or neither party).*
- ***Efficiency.*** *The client runs in time* $\mathrm{poly}(\lambda, |x|, m^*, \log t^*, k^*)$*, where* m^* *is the description length of the largest Turing machine* $\mathcal{M}$ *input by the client, and* t^* *is the largest step-size* t *input by the client, and* k^* *is the number of* Eval *phases that were executed. The server runs in time* $\mathrm{poly}(\lambda, |x|, m^*, t^*, k^*)$*.*

Note that the client's runtime is polynomial in $\log t^*$, which is much smaller than t^*, the number of timesteps that the machine runs for.

7.2 Construction

Overview

Imagine that x is a single bit. Then the ciphertext encrypting x is a quantum state of the following form:

$$\mathsf{FHE.Enc}[\mathsf{pk}, (S, x \oplus \langle \mathsf{v}, \mathbf{1}\rangle)],\ |S_{\mathsf{v},\mathsf{w}}\rangle \tag{8}$$

where $|S_{\mathsf{v},\mathsf{w}}\rangle$ is a subspace coset state. Essentially, x is masked with a uniformly random bit $\langle \mathsf{v}, \mathbf{1}\rangle$, and it is encrypted along with the information, S, that is needed to remove the mask.

Next, the client sends the ciphertext to the server and asks them to compute $\mathcal{M}^t(x)$ homomorphically. Since $(S, x \oplus \langle \mathsf{v}, \mathbf{1}\rangle)$ are encrypted using an FHE scheme, the server can *homomorphically* remove the mask, recover x, and compute $\mathcal{M}^t(x)$. The server does these computations in superposition over the quantum ciphertext without measuring anything. Therefore, their output is a superposition over encryptions of $\mathcal{M}^t(x)$. Then the server sends their registers to the client. The client checks that the server computed their output correctly and decrypts it to learn $y = \mathcal{M}^t(x)$. Here, the client measures y, but it is a gentle measurement. Because the client already checked that the server computed their output correctly, they will measure the correct value of y with overwhelming probability.

Finally, to delete the data, the client and server uncompute their previous computations, and then the server measures $|S_{\mathsf{v},\mathsf{w}}\rangle$ in the Hadamard basis. The value they measure is a vector $\widetilde{\mathsf{z}} \in S^\perp + \mathsf{w}$, which serves as the certificate of deletion. The reason why this "deletes" x is that by measuring the state in the Hadamard basis, the server destroys the information they have about v. Then x is statistically hidden, forever masked behind the bit $\langle \mathsf{v}, \mathbf{1}\rangle$.

Homomorphic Evaluation

The main goal of the protocol is to evaluate $\mathcal{M}^t$ homomorphically on an encryption of x. In this section, we will define ServerEval, the subprotocol that does this. It uses FHE.Eval to do the heavy lifting and applies a preprocessing function UnmaskInput to correctly format the input to FHE.Eval.

First, the encryption of x will be a ciphertext of the following form: let ℓ be the length of x, and for each bit $i \in [\ell]$, let $|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle$ be a subspace coset state. Let $x'_i = x_i \oplus \langle \mathsf{v}_i, \mathbf{1} \rangle$, and let $\mathsf{ct}_i^{(0)} = \mathsf{FHE.Enc}[\mathsf{pk}, (S_i, x'_i)]$. Then let the ciphertext be:

$$\left(\mathsf{ct}_i^{(0)},\ |(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\right)_{i\in[\ell]}$$

Also, since UnmaskInput and ServerEval are classical functions applied in superposition to the quantum ciphertext, we'll represent each state $|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle$ with a generic classical value $\mathsf{z}_i \in S_i + \mathsf{v}_i$ in the support of the superposition.

Next, UnmaskInput is defined below. It outputs x when the server and client are honest.

UnmaskInput

Inputs: $(S_i, x'_i, \mathsf{z}_i)_{i\in[\ell]}$

1. For each $i \in [\ell]$:
   (a) Compute the coset $\mathsf{v}'_i \in \mathsf{co}(S_i)$ that z_i belongs to.
   (b) Compute $x''_i = x'_i \oplus \langle \mathsf{v}'_i, \mathbf{1} \rangle$.
2. **Output** $x'' := (x''_1, \ldots, x''_\ell)$.

Note that when the server and client are honest, $\mathsf{v}_i = \mathsf{v}_i$, $x''_i = x_i$, and $x'' = x$.

Now we'll define ServerEval. It outputs an encryption of $\mathcal{M}^t(x)$ when the server and client are honest.

ServerEval

Inputs: $\mathsf{pk}, \mathcal{M}, t, (\mathsf{ct}_i^{(0)}, \mathsf{z}_i)_{i\in[\ell]}$:

1. Encrypt the z_is. For each $i \in [\ell]$:
$$\text{let } \mathsf{ct}_i^{(1)} = \mathsf{FHE.Enc}(\mathsf{pk}, \mathsf{z}_i)$$
2. Homomorphically compute UnmaskInput:
$$\mathsf{ct}^{(2)} = \mathsf{FHE.Eval}\left[\mathsf{pk}, \mathsf{UnmaskInput}, (\mathsf{ct}_i^{(0)}, \mathsf{ct}_i^{(1)})_{i\in[\ell]}\right]$$

3. Homomorphically compute $\mathcal{M}^t$. First compute $C_{\mathcal{M}^t}$, the circuit description of $\mathcal{M}^t$. Then compute:

$$\mathsf{ct}^{(3)} = \mathsf{FHE.Eval}\big(\mathsf{pk}, C_{\mathcal{M}^t}, \mathsf{ct}^{(2)}\big)$$

Finally, **output** $\mathsf{ct}^{(3)}$.

Note that when the server and client are honest, $(\mathsf{ct}_i^{(0)}, \mathsf{ct}_i^{(1)})_{i\in[\ell]}$ are an encryption of $(S_i, x'_i, \mathsf{z}_i)_{i\in[\ell]}$. Then $\mathsf{ct}^{(2)}$ is an encryption of x, and $\mathsf{ct}^{(3)}$ is an encryption of $\mathcal{M}^t(x)$.

Finally, let us define a Turing machine $\mathcal{M}'$ that runs ServerEval with some inputs hardwired.

$$\mathcal{M}' = \mathsf{ServerEval}\big[\mathsf{pk}, \mathcal{M}, t, (\mathsf{ct}_i^{(0)}, \cdot)_{i\in[\ell]}\big]$$

The only input to $\mathcal{M}'$ is the quantum part of the ciphertext: $\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[\ell]}$. The classical part is hardwired. Finally, $\mathcal{M}'$ computes an encryption of $\mathcal{M}^t(x)$ by running ServerEval on the ciphertext.

Full Construction

Here is the construction of the protocol for blind delegation with certified deletion ($\Pi_{\mathcal{F}_{BD}^{\mathsf{Del}}}$). It will call ServerEval (defined above) as a subroutine.

Setup

Inputs: $1^\lambda, x$

1. The client sets up an FHE scheme:
$$\mathsf{FHE.Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$$
2. The client encrypts x:
Let ℓ be the length of x, and let $n = 4\lambda$. Then, for each $i \in [\ell]$:
   (a) They sample a subspace S_i of dimension $n/2$ uniformly at random, along with two vectors $(\mathsf{v}_i, \mathsf{w}_i) \leftarrow \mathsf{co}(S_i) \times \mathsf{co}(S_i^\perp)$. Then they generate:
$$\mathsf{ct}_i^{(0)} = \mathsf{FHE.Enc}[\mathsf{pk}, (S_i, x_i \oplus \langle \mathsf{v}, \mathbf{1}\rangle)]$$
$$\mathsf{Z}_i = |(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle$$
   Some notation: Z_i is a quantum register, which holds the state $|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle$. Also let $\mathsf{Z} := (\mathsf{Z}_i)_{i\in[\ell]}$.
   (b) They sample a subspace T_i uniformly at random such that $S_i \leq T_i$ and $\dim(T_i) = 3n/4$ using the procedure specified in Definition 5.2. Let $\mathsf{u}_i \in \mathsf{co}(T_i)$ be the unique coset such that $S_i + \mathsf{v}_i \subset T_i + \mathsf{u}_i$.
3. The client sends register Z to the server. Then they store some information for later on the long-term secrets tape:
$$\mathbf{sec} = \mathsf{pk}, \mathsf{sk}, (S_i, T_i, \mathsf{u}_i, \mathsf{w}_i, \mathsf{ct}_i^{(0)})_{i\in[\ell]}$$

Eval is repeated an arbitrary number of times, once for each function $\mathcal{M}^t$ to be computed on x.

Eval

Inputs: $\mathcal{M}, t$

1. Client: The client sets up a SNARG scheme to certify the computation of $\mathcal{M}^t(x)$.
   (a) Let $\mathcal{M}' = \mathsf{ServerEval}\big[\mathsf{pk}, \mathcal{M}, t, (\mathsf{ct}_i^{(0)}, \cdot)_{i\in[\ell]}\big]$ (defined above). Recall that the only input to $\mathcal{M}'$ is the register Z, and the rest of the inputs are hardwired.
   (b) Next, the client computes $\ell' = \mathrm{poly}(\lambda, \ell, m)$, which upper bounds the input length of $\mathcal{M}'$, along with $t' = \mathrm{poly}(\lambda, \ell, t, m)$, which upper bounds the runtime of $\mathcal{M}'$. Then they compute:
   $$\mathsf{SNARG.Gen}(1^\lambda, \mathcal{M}', t', \ell') \to (\mathsf{SNARG\_pk}, \mathsf{SNARG\_vk})$$
   In our notation, the description of $\mathcal{M}'$ is included in the public key $\mathsf{SNARG\_pk}$.
   (c) Then the client sends the server $(\mathsf{SNARG\_pk}, \mathsf{SNARG\_vk})$.
2. Server:
   (a) The server applies $\mathcal{M}'$ in superposition to Z, while computing a SNARG certifying that the computation was done correctly. With our notation, that looks like this:
   $$\mathsf{Y}, \mathsf{P} \leftarrow \mathsf{SNARG.Prove}\big(\mathsf{SNARG\_pk}, \mathsf{Z}\big)$$
   where Y is the register containing the output of $\mathcal{M}'$, and P is the register containing the SNARG proof. They are entangled with each other and with Z.
   (b) The server sends registers $(\mathsf{Z}, \mathsf{Y}, \mathsf{P})$ to the client.
3. Client:
   (a) The client verifies the SNARG. First, they compute in superposition the function:
   $$\mathsf{SNARG.Verify}\big(\mathsf{SNARG\_vk}, \mathsf{Z}, \mathsf{Y}, \mathsf{P}\big)$$
   and then they measure the output bit. If the output is 1, they continue. Otherwise, they **output** abort and the protocol ends.
   (b) Next they verify that each $\mathsf{Z}_i \in T_i + \mathsf{u}_i$. Specifically, for each $i \in [\ell]$, they compute $P_{T_i+\mathsf{u}_i}(\mathsf{Z}_i)$ in superposition and measure the output. If for some i they measure 0, then they they **output** abort and the protocol ends. Otherwise they continue.
   (c) The client decrypts Y in superposition, by running $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{Y})$ in superposition and measuring the result, y. Then the client **outputs** y.
   (d) The client uncomputes the computations they performed (except measurements) and sends the registers $\mathsf{Z}, \mathsf{Y}, \mathsf{P}$ back to the server.
4. Server: The server uncomputes the computations they performed.

Del

1. Client: The client requests the deletion of their data by sending $\mathsf{Deletion\_Requested}$ to the server.
2. Server: For each $i \in [\ell]$, the server applies $H^{\otimes n}$ to Z_i and then measures the register to get the value $\widetilde{\mathsf{z}}_i$. They send each $\widetilde{\mathsf{z}}_i$ to the client.

3. Client: For each $i \in [\ell]$, the client checks that $P_{S_i^\perp + \mathsf{w}_i}(\widetilde{\mathsf{z}}_i) = 1$. If all checks pass, they **output** $\mathsf{DelOutcome} = 1$, and otherwise they output $\mathsf{DelOutcome} = 0$.

Lemma 7.4 (Correctness)**.** *If the server (as well as the client) is honest, then with overwhelming probability in λ, the following occurs: on every execution of* Eval*, the client outputs $y = \mathcal{M}^t(x)$, and* $\mathsf{DelOutcome} = 1$.

Proof. During Eval, $\mathcal{M}'$ is correctly computed with overwhelming probability. In that case, $\mathsf{Y} = \mathcal{M}'(\mathsf{Z})$. Also the Z register satisfies: $P_{S_i+\mathsf{v}_i}(\mathsf{Z}_i) = 1$ for all i. This implies that each classical value in Y's superposition is an encryption of $\mathcal{M}^t(x)$. This is because for each classical value z_i in Z_i's superposition: $\mathsf{z}_i \in S_i + \mathsf{v}_i$. Then $\mathsf{UnmaskInput}[(S_i, x_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle, \mathsf{z}_i)_{i\in[\ell]}]$ outputs x, and $\mathcal{M}'[(\mathsf{z}_i)_{i\in[\ell]}] = \mathsf{ServerEval}(\mathsf{pk}, \mathcal{M}, t, (\mathsf{ct}_i^{(0)}, \mathsf{z}_i)_{i\in[\ell]})$ outputs an encryption of $\mathcal{M}^t(x)$.

Next, the measurements in Eval produce a particular outcome with overwhelming probability, so they change the state of the system by a negligible amount (measured in trace distance). Then at the start of Del, the Z register is negligibly close to what it was originally: for each i, $\mathsf{Z}_i = |(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle$. Then the server measures a collection of vectors $\widetilde{\mathsf{z}}_i \in S_i^\perp + \mathsf{w}_i$, which the client accepts. □

7.3 Efficiency

Theorem 7.5. *The construction of blind delegation with certified deletion satisfies efficiency (Definition 7.3).*

Proof. As before, let k^* be the number of times Eval is run, let m^* be the description length of the largest Turing machine $\mathcal{M}$ input by the client, and let t^* be the largest step-size t input by the client. Next, note that the entire protocol runs in time $\mathrm{poly}(\lambda, \ell, m^*, t^*, k^*)$ because every step runs in time polynomial in the input length, and the inputs to each phase have length $\mathrm{poly}(\lambda, \ell, m^*, t^*)$. Now it remains to show that the client's computations in the protocol take time $\mathrm{poly}(\lambda, \ell, m^*, \log t^*, k^*)$.

The input to Setup has length $\lambda + \ell$, and Setup runs in time $\mathrm{poly}(\lambda, \ell)$.

Next, Eval is run k^* times, and we'll show that each time, the client's computations take time $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$. In Eval step 1, the description of $\mathcal{M}', t'$ takes space $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$, so SNARG.Gen takes time $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$. Also, the time it takes to compute the description of $\mathcal{M}', t'$ is $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$. In Eval step 3, the client runs SNARG.Verify, which takes time $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$ (Definition 4.12). The rest of Eval step 3 deals with variables that take $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$ space, so it runs in time $\mathrm{poly}(\lambda, \ell, m^*, \log t^*)$.

Finally, the client's computations in Del take time $\mathrm{poly}(\lambda, \ell)$. □

7.4 Security

We will use the following simulator $\mathcal{S}_\lambda$ to prove both computational security and certified deletion. It will simulate the real-world protocol with input $x' = 0^l$ in place of x.

Definition 7.6 (The Simulator)**.** *The simulator is a collection of QPT functions $\mathcal{S}_\lambda = (\mathcal{S}_{\lambda,1}, \ldots, \mathcal{S}_{\lambda,k})$ with the following behavior:*

1. *$\mathcal{S}_{\lambda,1}$ runs* $\mathsf{Setup}(1^\lambda, 0^l)$*. This includes initializing a tape* **sec***.*
2. *For each round of* Eval*, the simulator receives $(\mathcal{M}, t)$ from the ideal functionality. Then they run a modified version of* $\mathsf{Eval}(\mathcal{M}, t)$*, which is defined as follows. They run the honest client protocol and*

use $\mathcal{A}_\lambda$ to simulate the server. If the simulated client aborts in steps 3a or 3b, then the simulator outputs abort. *If the simulator reaches step 3c, they skip that step and instead just output* deliver.

3. *When they receive* Deletion_Requested, *the simulator runs* Del, *simulating both the honest client and the adversarial server $\mathcal{A}_\lambda$. If the simulated client outputs* $\mathsf{DelOutcome} = 1$, *then the simulator outputs* Deletion_Confirmed. *Finally, in the certified deletion definition, they also output their tape* **sec** *when* $\mathsf{DelOutcome} = 1$.

Theorem 7.7. *The construction of blind delegation with certified deletion satisfies computational security (Definition 7.3).*

Proof. The following sequence of hybrids transforms the real-world protocol to the ideal-world protocol. We'll only say how each hybrid differs from the one before it:

- $\mathcal{H}_0$ is $\Pi_{\mathcal{F}_{BD}^{\mathsf{Del}}}(\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle)$, the real-world protocol.
- $\mathcal{H}_1$: If the client reaches Eval step 3c (where they decrypt the ciphertext), they skip that step and instead compute and output $y = \mathcal{M}^t(x)$.
- $\mathcal{H}_2$: Instead of running $\mathsf{Setup}(1^\lambda, x)$, the client runs $\mathsf{Setup}(1^\lambda, 0^\ell)$.

First, the output of $\mathcal{H}_0$ is statistically close to the output of $\mathcal{H}_1$: $\mathsf{TD}[\mathcal{H}_0, \mathcal{H}_1] = \mathrm{negl}(\lambda)$. This is because on any execution of Eval the following occurs with overwhelming probability: either the client aborts or they measure $y = \mathcal{M}^t(x)$ (Lemma 7.9). Next, consider two cases. First, for the given $(\mathcal{Z}_\lambda, \mathcal{A}_\lambda, |\psi_\lambda\rangle)$ and the given phase of Eval, the probability that the client aborts is overwhelming. If the client aborts, then they never reach step 3c. Therefore, the output on this round is statistically close between $\mathcal{H}_0$ and $\mathcal{H}_1$. Second, if the probability of aborting on the given phase of Eval is not overwhelming, then given that the client does not abort, they measure $y = \mathcal{M}^t(x)$ with overwhelming probability. This is a gentle measurement; because it produces a particular outcome with overwhelming probability, the measurement changes the quantum state by a negligible amount (measured in trace distance). So even though the client omits this measurement in $\mathcal{H}_1$, this changes the outcome of the hybrid by a negligible amount.

Second, $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable to any QPT distinguisher $\mathcal{D}_\lambda$, because FHE is semantically secure. Eval step 3c is the only place in the real-world protocol where the FHE decryption key sk is used. Since $\mathcal{H}_1$ and $\mathcal{H}_2$ omit this step, both hybrids can be simulated without sk, which lets us reduce to the semantic security game.

In the semantic security game, the challenger samples $(\mathsf{pk}, \mathsf{sk})$ by running $\mathsf{FHE.Gen}(1^\lambda)$, which is the same as Setup step 1 in our construction. Then they send pk to the semantic security adversary, and the adversary simulates the rest of $\mathcal{H}_1$ or $\mathcal{H}_2$. In Setup step 2a, when they need to encrypt a message, the adversary sends the challenger two options, $(S_i, x_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle)_{i\in[\ell]}$ or $(S_i, 0 \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle)_{i\in[\ell]}$. This allows the adversary to simulate $\mathcal{H}_1$ or $\mathcal{H}_2$, depending on which message the challenger encrypted. Then they use the distinguisher $\mathcal{D}_\lambda$ to distinguish which of the two hybrids they simulated. By the semantic security of FHE, the distinguisher must have negligible advantage.

Third, note that the output of $\mathcal{H}_2$ has the same distribution as $\widetilde{\Pi}_{\mathcal{F}_{BD}^{\mathsf{Del}}}(\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle)$, the ideal-world protocol that uses the simulator from Definition 7.6. The inner workings of $\mathcal{H}_2$ may differ from the ideal-world protocol, but their output distributions are the same.

Therefore any QPT distinguisher has $\mathrm{negl}(\lambda)$ advantage at distinguishing $\Pi_{\mathcal{F}_{BD}^{\mathsf{Del}}}(\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle)$ and $\widetilde{\Pi}_{\mathcal{F}_{BD}^{\mathsf{Del}}}(\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle)$. □

Theorem 7.8. *The construction of blind delegation with certified deletion satisfies certified deletion (Definition 7.3).*

Proof. Recall that $\Pi^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}_{BD}}[\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ comprises the output of $\Pi_{\mathcal{F}^{\mathsf{Del}}_{BD}}[\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ as well as the client's tape sec when $\mathsf{DelOutcome} = 1$, and it comprises only $\perp$ when $\mathsf{DelOutcome} = 0$. In the ideal world, $\widetilde{\Pi}^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}_{BD}}[\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$ is defined analogously, except the simulator, not the ideal client, outputs $\mathbf{sec}$. We will show that these two distributions are statistically close.

The following sequence of hybrids transforms the real-world protocol to the ideal-world protocol. We'll only say how each hybrid differs from the one before it.

- $\mathcal{H}_0$ is $\Pi^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}_{BD}}[\mathcal{A}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$
- $\mathcal{H}_1$: If the client reaches Eval step 3c, they skip it and simply output $y = \mathcal{M}^t(x)$.

For each $i \in [\ell]$:

- $\mathcal{H}_{i+1}$: Let $x^{(i)} = (0^i, x_{[i+1,\ell]})$. That's to say: the first i bits are 0 and the rest match x. Then the client runs $\mathsf{Setup}(1^\lambda, x^{(i)})$.
- $\mathcal{H}_{\ell+1}$: The client runs $\mathsf{Setup}(1^\lambda, 0^\ell)$.

First, the outputs of $\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically close: $\mathsf{TD}[\mathcal{H}_0, \mathcal{H}_1] = \mathrm{negl}(\lambda)$. This follows from the same argument we used in the proof of computational security (Theorem 7.7).

Second, $\mathsf{TD}[\mathcal{H}_1, \mathcal{H}_2] = \mathrm{negl}(\lambda)$. We'll show this with a reduction to the certified deletion game in Definition 6.1. Assume toward contradiction that there is an adversarial server for which $\mathsf{TD}[\mathcal{H}_1, \mathcal{H}_2]$ is non-negligible. Then there is an adversary for Definition 6.1 that simulates $\mathcal{H}_1$ or $\mathcal{H}_2$ and achieves non-negligible advantage in Theorem 6.5.

Let's fix some notation. The *adversary* and *challenger* are the players in Definition 6.1, and the *server* and *client* are the players in $\mathcal{H}_1$ or $\mathcal{H}_2$. The input to the game in Definition 6.1 is bit b that corresponds to x_1. If $b = 0$, we will end up simulating $\mathcal{H}_2$, and if $b = 1$, we will end up simulating $\mathcal{H}_1$ (we can assume that $x_1 = 1$ in $\mathcal{H}_1$ because otherwise, $\mathcal{H}_1$ and $\mathcal{H}_2$ would be identical).

Now here is what the game in Definition 6.1 looks like with the adversary that simulates $\mathcal{H}_1$ or $\mathcal{H}_2$:

$\underline{\mathsf{EXP}^*(b)}$:

1. The challenger receives b as input and sets $x_1 = b$. Then they set up the encryption scheme:

$$\mathsf{FHE.Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$$

   Then they sample subspaces (S_1, T_1) and vectors $(\mathsf{u}_1, \mathsf{v}_1, \mathsf{w}_1)$ from the specified distribution. Then they compute $\mathsf{Z}_1 = |(S_1)_{\mathsf{v}_1,\mathsf{w}_1}\rangle$, $x'_1 = x_1 \oplus \langle \mathsf{v}_1, 1\rangle$ and $\mathsf{ct}_1^{(0)} = \mathsf{FHE.Enc}[\mathsf{pk}, (S_1, x'_1)]$. Finally they send the adversary

$$\mathsf{Z}_1, \mathsf{ct}_1^{(0)}, \mathsf{pk}, T_1, \mathsf{u}_1$$

2. (a) The adversary simulates the rest of Setup, from step 2 onward. For every $i \in \{2, \ldots, \ell\}$, they sample $(S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \mathsf{w}_i)$, and they compute x'_i and $\mathsf{ct}_i^{(0)}$. Finally, they initialize the tape $\mathbf{sec}$ as follows:

$$\mathbf{sec} = \mathsf{pk}, T_1, \mathsf{u}_1, \mathsf{ct}_1^{(0)}, (S_i, T_i, \mathsf{u}_i, \mathsf{w}_i, \mathsf{ct}_i^{(0)})_{i \geq 2}$$

$\mathbf{sec}$ includes all the expected variables except $(\mathsf{sk}, S_1, \mathsf{w}_1)$, since the adversary doesn't know them.

(b) The adversary simulates Eval by executing the client and server's algorithms, except they skip step 3c.

(c) The adversary simulates Del steps 1 and 2. In step 3, they must verify the deletion certificate. First they verify that for all $i \in \{2, \ldots, \ell\}$, $P_{S_i^\perp + \mathsf{w}_i}(\widetilde{\mathsf{z}}_i) = 1$. If any check fails, they set $\rho = \perp$. Otherwise, they let ρ include $\mathbf{sec}$, any outputs of the protocol so far, and any state held by the server.

Finally, the adversary sends $(\widetilde{\mathsf{z}}_1, \rho)$ to the challenger.

3. The challenger checks whether $P_{S_1^\perp + \mathsf{w}_1}(\widetilde{\mathsf{z}}_1) = 1$. If so they output $(\rho, S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, x'_1)$ and if not, they output $\perp$.

We can show that $\mathsf{TD}[\mathsf{EXP}^*(0), \mathsf{EXP}^*(1)] = \mathrm{negl}(\lambda)$ because the auxiliary information sent to the adversary is subspace-hiding. The auxiliary information of the adversary's challenge is

$$\mathsf{FHE.Enc}(\mathsf{pk}, [S_1, x'_1)], \mathsf{pk}, T_1, \mathsf{u}_1$$

This is computationally indistinguishable from $[\mathsf{FHE.Enc}(0^*), \mathsf{pk}, T_1, \mathsf{u}_1]$, where 0^* is the same length as (S_1, x'_1). That is to say, the auxiliary information is computationally indistinguishable from something that does not depend on S_1 or w_1. Therefore it satisfies the notion of subpsace hiding defined in Definition 6.4. Then Theorem 6.5 says that $\mathsf{TD}[\mathsf{EXP}^*(0), \mathsf{EXP}^*(1)] = \mathrm{negl}(\lambda)$.

Next, steps 1 and 2 of EXP^* correctly simulate $\mathcal{H}_1$ or $\mathcal{H}_2$ up through Del step 2, except that $\mathbf{sec}$ is missing $(\mathsf{sk}, S_1, \mathsf{w}_1)$. However, in step 3 of EXP^*, the challenger appends (S_1, w_1) to the output if $P_{S_1^\perp + \mathsf{w}_1}(\widetilde{\mathsf{z}}_1) = 1$. Furthermore, given pk, which is included in the output, an unbounded machine can sample sk from the correct distribution. (Because we're using public-key encryption, all ciphertexts were generated using pk, not sk, so the distribution of sk, even given the ciphertexts, just depends on pk).

Therefore $\mathsf{TD}\big([\mathsf{EXP}^*(0), \mathsf{sk}], [\mathsf{EXP}^*(1), \mathsf{sk}]\big) = \mathsf{TD}[\mathsf{EXP}^*(0), \mathsf{EXP}^*(1)]$. Since $(\mathsf{EXP}^*, \mathsf{sk})$ includes every variable output by $\mathcal{H}_1$ or $\mathcal{H}_2$, $\mathsf{TD}[\mathcal{H}_1, \mathcal{H}_2]$ is $\mathrm{negl}(\lambda)$.

By the same reasoning, we can show that for any adjacent hybrids $(\mathcal{H}_i, \mathcal{H}_{i+1})$, where $i \in [\ell]$, $\mathsf{TD}[\mathcal{H}_i, \mathcal{H}_{i+1}] = \mathrm{negl}(\lambda)$. Therefore, $\mathsf{TD}[\mathcal{H}_0, \mathcal{H}_{\ell+1}] = \mathrm{negl}(\lambda)$.

Finally, note that the output of $\mathcal{H}_{\ell+1}$ is the same as the output of $\widetilde{\Pi}^{\mathsf{DelOutcome}=1}_{\mathcal{F}^{\mathsf{Del}}_{BD}}[\mathcal{S}_\lambda, \mathcal{Z}_\lambda, |\psi_\lambda\rangle]$. □

Lemma 7.9 (Integrity)**.** *In the real-world protocol, the probability is overwhelming in λ that on every execution of* Eval*, the client aborts or outputs $y = \mathcal{M}^t(x)$.*

Proof. Consider the following sequence of hybrids. We'll only state how each hybrid differs from the one before it.

- $\mathcal{H}_0$: Run the real-world protocol. Output 1 if on every execution of Eval, the client aborts or outputs $y = \mathcal{M}^t(x)$. Otherwise output 0.

For each $i \in [\ell]$:

- $\mathcal{H}_i$: Whenever Eval step 3b is run, the client checks that $P_{S_i+\mathsf{v}_i}(\mathsf{Z}_i) = 1$ (instead of checking that $P_{T_i+\mathsf{u}_i}(\mathsf{Z}_i) = 1$).

The probability of outputting 1 is negligibly close in $\mathcal{H}_0$ and $\mathcal{H}_1$. We can show this by reducing to the subspace-hiding game of Theorem 4.11. The server's inputs depend on S_1 and v_1 but do not otherwise depend on T_1 or u_1. From the server's point of view, T_1 is a uniformly random superspace of S_1 such that $\dim(T_1) = 3n/4$, and $\mathsf{u}_1 \in \mathsf{co}(T_1)$ is the unique coset such that $S_1 + \mathsf{v}_1 \subset T_1 + \mathsf{u}_1$. The reduction to the subspace-hiding game is the following: the adversary for the subspace-hiding game choooses a random (S_1, v_1), sends them to the challenger, and then simulates $\mathcal{H}_0$ or $\mathcal{H}_1$. When they need to execute Eval step 3b, they query the challenger's oracle, which implements $P_{T_1+\mathsf{u}_1}$ or $P_{S_1+\mathsf{v}_1}$. Depending on which function the challenger chose to implement, the adversary ends up simulating $\mathcal{H}_0$ or $\mathcal{H}_1$. Finally, the adversary outputs the whatever bit the hybrid outputs. By Theorem 4.11, the probability of outputting 1 in $\mathcal{H}_0$ and $\mathcal{H}_1$ must be negligibly close.

By the same argument, we can show that for any adjacent hybrids $(\mathcal{H}_i, \mathcal{H}_{i+1})$, the probability of outputting 1 is negligibly close. Since there are polynomially-many hybrids, then the probability of outputting 1 is negligibly close between $\mathcal{H}_0$ and $\mathcal{H}_\ell$.

Next, we'll show that in $\mathcal{H}_\ell$, the probability of outputting 1 is overwhelming. First, by the soundness of the SNARG, the following occurs with overwhelming probability: SNARG.Verify rejects or registers (Z, Y) satisfy $\mathsf{Y} = \mathcal{M}'(\mathsf{Z})$. If the client does not abort, then right before they measure y, register Z satisfies: $P_{S_i+\mathsf{v}_i}(\mathsf{Z}_i) = 1$ for all i. We already showed that if register Z satsifies that condition, and $\mathsf{Y} = \mathcal{M}'(\mathsf{Z})$, then decrypting register Y and measuring the result produces $y = \mathcal{M}^t(x)$. Therefore, $\mathcal{H}_\ell$ outputs 1 with overwhelming probability, and this implies that $\mathcal{H}_0$ does as well. □

8 Obfuscation with Certified Deletion

8.1 Definitions

Definition 8.1 (Differing inputs obfuscation with certified deletion)**.** *An differing inputs obfuscation scheme with (publicly-verifiable) certified deletion for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ has the following syntax.*

- $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C) \to |\widetilde{C}\rangle, \mathsf{vk}$*: The obfuscation algorithm takes as input the security parameter 1^λ, a circuit $C \in \mathcal{C}_\lambda$ and outputs a (quantum) obfuscated program $|\widetilde{C}\rangle$ and a verification key* vk*.*
- $\mathsf{Eval}(|\widetilde{C}\rangle, x) \to y$*: The evaluation algorithm takes as input the obfuscated program $|\widetilde{C}\rangle$ and a (classical) input x, and outputs a (classical) y.*
- $\mathsf{Del}(|\widetilde{C}\rangle) \to \mathsf{cert}$*: The deletion algorithm takes as input the obfuscated program $|\widetilde{C}\rangle$ and outputs a deletion certificate* cert*.*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) \to \{\top, \bot\}$*: The verification algorithm takes as input the verification key and a deletion certificate and outputs either $\top$ or $\bot$.*

It should satisfy the following properties.

- ***Functionality preservation.*** *For all $\lambda \in \mathbb{N}$, all $C \in \mathcal{C}_\lambda$, and all inputs x,*

$$\Pr[\mathsf{Eval}(|\widetilde{C}\rangle, x) = C(x) : |\widetilde{C}\rangle, \mathsf{vk} \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C)] = 1.$$

We remark that even if the description of Eval *involves measurements, the above correctness guarantee implies that the obfuscated state $|\widetilde{C}\rangle$ can be reused an arbitrary number of times, by implementing* Eval *coherently and just measuring the output bit.*

- ***Correctness of deletion.*** *For all sequence of circuits $\{C_\lambda \in \mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top : \begin{array}{r} |\widetilde{C}\rangle, \mathsf{vk} \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C_\lambda) \\ \mathsf{cert} \leftarrow \mathsf{Del}(|\widetilde{C}\rangle) \end{array}\right] = 1 - \mathrm{negl}(\lambda).$$

- ***Computational security.*** *Let $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a differing inputs circuits family associated with an efficiently sampleable distribution family $\{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$. Then for all QPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$\left|\begin{array}{c} \Pr[\mathcal{A}(C_0, C_1, \mathsf{aux}, \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^n, C_0))) = 1 : (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_\lambda] \\ -\Pr[\mathcal{A}(C_0, C_1, \mathsf{aux}, \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^n, C_1))) = 1 : (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_\lambda] \end{array}\right| = \mathrm{negl}(\lambda)$$

- ***Certified everlasting security.*** *Let $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a differing inputs circuits family associated with a sampler $\{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$. For all QPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$\mathsf{TD}\left(\mathsf{EV}\text{-}\mathsf{EXP}(0, \mathcal{A}_\lambda), \mathsf{EV}\text{-}\mathsf{EXP}(1, \mathcal{A}_\lambda)\right) = \mathrm{negl}(\lambda),$$

where the experiment $\mathsf{EV}\text{-}\mathsf{EXP}(b, \mathcal{A})$ is defined as follows.

  - *Sample $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_\lambda$.*[15] *Sample $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C_b)$ and initialize $\mathcal{A}$ with $(C_0, C_1, \mathsf{aux}, |\widetilde{C}\rangle)$.*
  - *Parse $\mathcal{A}$'s output as a deletion certificate* cert *and a left-over quantum state ρ.*
  - *If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top$ then output ρ, and otherwise output $\bot$.*

We say it has public verifiability if this holds even when $\mathcal{A}$ is also initialized with vk*.*

Note that we require that even an unbounded adversary cannot tell the leftover state from $\mathsf{EV}\text{-}\mathsf{EXP}(0, \mathcal{A}_\lambda)$ and $\mathsf{EV}\text{-}\mathsf{EXP}(1, \mathcal{A}_\lambda)$ apart. Such an adversary can compute the differing inputs for themselves, since $\mathcal{A}$ was given the *classical* descriptions of C_0 and C_1. Thus, even an unbounded adversary cannot evaluate the obfuscated program on the differing inputs after deletion.

In this work, we consider the case of differing inputs circuits families with a polynomial number of differing inputs. One notable special case is the case of zero differing inputs. We call this case *indistinguishability obfuscation with certified deletion* ($\mathsf{i}\mathcal{O}$-CD). It guarantees that for any two functionally equivalent circuits C_0 and C_1, $\mathsf{i}\mathcal{O}\text{-}\mathsf{CD}(C_0)$ is statistically close to $\mathsf{i}\mathcal{O}\text{-}\mathsf{CD}(C_1)$ after they have been deleted.[16]

[15] Recall that we restrict aux to be classical. Our definitions and constructions also extend to the case of a quantum aux, but require explicitly assuming a suitable $\mathsf{di}\mathcal{O}$, as it is not clearly implied by $\mathsf{i}\mathcal{O}$. However, a classical aux is sufficient for our applications.

[16] Intriguingly, $\mathsf{i}\mathcal{O}$-CD does not imply $\mathsf{di}\mathcal{O}$-CD for a polynomial number of differing inputs, unlike their counterparts which do not support deletion. In the latter case, one can use a distinguisher for $\mathsf{di}\mathcal{O}$ to find the differing inputs for a given pair of circuits C_0, C_1 in polynomial time. This would violate the properties of a differing inputs circuit class. However, the distinguisher for $\mathsf{di}\mathcal{O}$-CD only manages to distinguish after deletion. Since it is allowed unbounded computation after deletion, it can find the differing inputs at this point regardless of whether it can successfully distinguish the obfuscation.

Nested differing inputs. We also introduce a new circuit class called *nested* differing inputs circuits, along with the corresponding security property. This property will allow generalizing the classical technique of wrapping a differing inputs obfuscation inside another indistinguishability obfuscator with a more general functionality. Since the functionality of the outer $\mathsf{i}\mathcal{O}$ may take in a larger input than the inner functionality, and may query the inner differing inputs obfuscation and use the output arbitrarily, there may be an exponential number of differing inputs.[17] As long as the adversary cannot find a differing input when given the description of the outer functionality, the inner functionality can still be switched according to the security of differing inputs obfuscation for a polynomial number of differing inputs. This technique allows additional versatility when using differing inputs obfuscation.

Unfortunately, wrapping a quantum program in a classical indistinguishability obfuscation is not well-defined. Therefore, we explicitly build this technique into the properties of $\mathsf{di}\mathcal{O}$-CD.

Definition 8.2 (Nested Differing Inputs Circuits)**.** *A nested differing inputs circuits family $\mathcal{C}'$ is defined by a circuit C_f and a differing inputs circuit family $\mathcal{C}$ with a distribution $\mathcal{D}_{\mathcal{C}}$. It consists of the circuits $C_f \circ C_{\mathcal{C}}$ for $C_{\mathcal{C}} \in \mathcal{C}$, which take as input bitstrings x and y then output $\Pi_f(x, y, C_{\mathcal{C}}(y))$. It is associated with a distribution $\mathcal{D}'$. Samples from $\mathcal{D}'$ are obtained by sampling $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_{\mathcal{C}}$ and outputting $(C_f \circ C_0, C_f \circ C_1, \mathsf{aux})$.*

y^* is a *nested* differing input for $(\Pi_f \circ \Pi_0, \Pi_f \circ \Pi_1, \mathsf{aux}) \leftarrow \mathcal{D}'$ if it is a differing input for (Π_0, Π_1). Note that two circuits $\Pi_f \circ \Pi_0$ and $\Pi_f \circ \Pi_1$ may have an exponential number of differing inputs even though they only have polynomially many nested differing inputs. However, all differing inputs are of the form (x, y^*) where y^* is a nested differing input.

Deletion security for nested differing inputs circuit families using the same $\mathsf{di}\mathcal{O}$-CD security game. We say a $\mathsf{di}\mathcal{O}$-CD scheme has deletion security for nested differing inputs circuits if it has deletion security for all nested differing inputs circuits families. We refer to such a scheme as a *nested* differing inputs obfuscation with certified deletion.

Succinct indistinguishability obfuscation. Next, we define *succinct* indistinguishability obfuscation, otherwise known as $\mathsf{i}\mathcal{O}$ for Turing machines. We generally parameterize a Turing machine M with a step-size t and an input length n. Given M, t, and an $x \in \{0,1\}^n$, we define $M^t(x)$ to be the value written on the output tape of M after taking x as input and running for t steps, if such a value exists. Otherwise, it is defined to be $\perp$.

Definition 8.3 (Succinct indistinguishability obfuscation with certified deletion)**.** *A succinct indistinguishability obfuscation scheme with (publicly-verifiable) certified deletion for a class of Turing machines $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ has the following syntax.*

- $\mathsf{si}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M, t, n) \to \left|\widetilde{M}\right\rangle, \mathsf{vk}$*: The obfuscation algorithm takes as input the security parameter 1^λ, the description of a Turing machine $M \in \mathcal{M}_\lambda$, a step-size t, and an input length n, and outputs a (quantum) obfuscated Turing machine $\left|\widetilde{M}\right\rangle$ and a verification key vk.*
- $\mathsf{Eval}(\left|\widetilde{M}\right\rangle, x) \to y$*: The evaluation algorithm takes as input the obfuscated program $\left|\widetilde{M}\right\rangle$ and a (classical) input x, and outputs a (classical) y.*

[17] For example, say the outer functionality takes as input (x, y), then ignores x and outputs $f(y)$, where f is the inner functionality.

- $\mathsf{Del}(|\widetilde{M}\rangle) \to \mathsf{cert}$*: The deletion algorithm takes as input the obfuscated program* $|\widetilde{M}\rangle$ *and outputs a deletion certificate* cert*.*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) \to \{\top, \bot\}$*: The verification algorithm takes as input the verification key and a deletion certificate and outputs either* $\top$ *or* $\bot$*.*

It should satisfy the following properties.

- ***Functionality preservation.*** *For all* $\lambda \in \mathbb{N}$*, all* $M \in \mathcal{M}_\lambda$*, all* $x \in \{0,1\}^n$*, and all* t*,*

$$\Pr[\mathsf{Eval}(|\widetilde{M}\rangle, x) = M^t(x) : |\widetilde{M}\rangle, \mathsf{vk} \leftarrow \mathsf{si}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M, t, n)] = 1.$$

  We remark that even if the description of Eval *involves measurements, the above correctness guarantee implies that the obfuscated state* $|\widetilde{M}\rangle$ *can be reused an arbitrary number of times, by implementing* Eval *coherently and just measuring the output bit.*

- ***Correctness of deletion.*** *For all sequences of Turing machines* $\{M_\lambda \in \mathcal{M}_\lambda, t_\lambda, n_\lambda\}_{\lambda\in\mathbb{N}}$*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top : \begin{array}{r} |\widetilde{M}\rangle, \mathsf{vk} \leftarrow \mathsf{si}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M_\lambda, t_\lambda, n_\lambda) \\ \mathsf{cert} \leftarrow \mathsf{Del}(|\widetilde{M}\rangle) \end{array}\right] = 1 - \mathrm{negl}(\lambda).$$

- ***Computational security.*** *For all sequences of functionally equivalent Turing machines* $\{M_{0,\lambda}, M_{1,\lambda}, t_\lambda, n_\lambda\}_{\lambda\in\mathbb{N}}$*, meaning that for all* $\lambda \in \mathbb{N}$*, and* $x \in \{0,1\}^{n(\lambda)}$*,* $M_{0,\lambda}^{t_\lambda}(x) = M_{1,\lambda}^{t_\lambda}(x)$*, and all QPT adversaries* $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$*,*

$$|\Pr[\mathcal{A}_\lambda(\mathsf{si}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M_{0,\lambda}, t_\lambda, n_\lambda)) = 1] - \Pr[\mathcal{A}_\lambda(\mathsf{si}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M_{1,\lambda}, t_\lambda, n_\lambda)) = 1]| = \mathrm{negl}(\lambda).$$

- ***Certified everlasting security.*** *For all sequences of functionally equivalent Turing machines* $\{M_{0,\lambda}, M_{1,\lambda}, t_\lambda, n_\lambda\}_{\lambda\in\mathbb{N}}$ *and all QPT adversaries* $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$*,*

$$\mathsf{TD}\left(\mathsf{EV}\text{-}\mathsf{EXP}(M_{0,\lambda}, t_\lambda, n_\lambda, \mathcal{A}_\lambda), \mathsf{EV}\text{-}\mathsf{EXP}(M_{1,\lambda}, t_\lambda, n_\lambda, \mathcal{A}_\lambda)\right) = \mathrm{negl}(\lambda),$$

  where, given a Turing machine (M, t, n) *and an adversary* $\mathcal{A}$*, the experiment* $\mathsf{EV}\text{-}\mathsf{EXP}(M, t, n, \mathcal{A})$ *is defined as follows.*

  - *Sample* $|\widetilde{M}\rangle, \mathsf{vk} \leftarrow \mathsf{si}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M, t, n)$ *and initialize* $\mathcal{A}$ *with* $(|\widetilde{M}\rangle, \mathsf{vk})$*.*
  - *Parse* $\mathcal{A}$*'s output as a deletion certificate* cert *and a left-over quantum state* ρ*.*
  - *If* $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top$ *then output* ρ*, and otherwise output* $\bot$*.*

- ***Succinctness.*** *The running time of* $\mathsf{si}\mathcal{O}\text{-}\mathsf{CD}$ *must be* $\mathrm{poly}(\lambda, |M|, \log t, n)$*.*

8.2 Construction

We provide a construction of $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}$ for circuits with polynomially-many differing inputs from post-quantum $\mathsf{i}\mathcal{O}$ (and one-way functions). The construction consists of two pieces. First, it contains the quantum part of a ciphertext which can be certifiably deleted. This ciphertext can be decrypted by measuring it in the computational basis then doing classical computation. Second, it contains the obfuscation of a classical program which has the classical part of the ciphertext hard-coded. This program takes as input a supposed measurement of the ciphertext along with the input x. If the measurement is valid, then the program decrypts it to obtain two circuits, one of which is evaluated on the input x.

diO-CD Classical Program

Hard-Coded Values. Subspace and offset tuples $\{S_i, T_i, \mathsf{u}_i\}_{i\in[2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$.

Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

1: **if** $\mathsf{t}_i \in T_i + \mathsf{u}_i$ for every $i \in [2\ell+1]$ **then**
2: Let $\mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to, and define $c'_i := \widetilde{c}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
3: Parse $c'_1, \ldots, c'_{2\ell+1}$ as (b, C_0, C_1), where C_0 and C_1 are circuits with description length ℓ. Output $C_b(x)$.

Differing Inputs Obfuscation with Certified Deletion

$\mathsf{diO\text{-}CD}(1^\lambda, C)$**:**

1: Let $n = 4\lambda$, let $\ell = |C|$, and for each $i \in [2\ell+1]$, sample $S_i < T_i < \mathbb{F}_2^n$ such that $\dim(S_i) = n/2$ and $\dim(T_i) = 3n/4$, sample $\mathsf{v}_i, \mathsf{w}_i \leftarrow \mathsf{co}(S_i) \times \mathsf{co}(S_i^\perp)$, and let u_i be the coset of T_i that v_i belongs to.
2: Let $c := (0, C, 0^\ell)$, and for all $i \in [2\ell+1]$, define $\widetilde{c}_i := c_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle$. Define $\widetilde{c} := (\widetilde{c}_1, \ldots, \widetilde{c}_{2\ell+1})$.
3: Compute the indistinguishability obfuscation $\mathsf{iO}_{\mathsf{class}}$ of the $\mathsf{diO\text{-}CD}$ classical program using hard-coded values $\{S_i, T_i, \mathsf{u}_i\}_{i\in[2\ell+1]}$ and $\widetilde{c}$.
4: Output

$$|\widetilde{C}\rangle := \left(\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]},\ \mathsf{iO}_{\mathsf{class}}\right), \quad \mathsf{vk} := \left\{\mathsf{iO}\left(P_{S_i^\perp + \mathsf{w}_i}\right)\right\}_{i\in[2\ell+1]}.$$

$\mathsf{Eval}(|\widetilde{C}\rangle, x)$**:**

1: Measure $\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]}$ in the computational basis to obtain vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$.[a]
2: Run $\mathsf{iO}_{\mathsf{class}}$ on input $(\{\mathsf{t}_i\}_{i\in[2\ell+1]}, x)$ to obtain y, then output y.

$\mathsf{Del}(|\widetilde{C}\rangle)$**:** Measure $\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]}$ in the Hadamard basis to obtain vectors $\{\mathsf{z}_i\}_{i\in[2\ell+1]}$, and output $\mathsf{cert} := \{\mathsf{z}_i\}_{i\in[2\ell+1]}$.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$**:** If $\mathsf{iO}\left(P_{S_i^\perp + \mathsf{w}_i}\right)(\mathsf{z}_i) = 1$ for all $i \in [2\ell+1]$ then output $\top$ and otherwise output $\bot$.

[a] As remarked in Definition 8.1, one can always perform this coherently and preserve the ability to run Eval on mutiple inputs.

Theorem 8.4. *Assuming post-quantum indistinguishability obfuscation and one-way functions, there exists differing inputs obfuscation with (publicly-verifiable) certified deletion for polynomially many differing inputs (Definition 8.1).*

Proof. First, we note that (perfect) functionality preservation and correctness of deletion (with some negligible error) are immediate from the scheme. Thus, it remains to show computational security and certified everlasting security.

Computational security. Consider two differing inputs circuits $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_\lambda$. We will switch an obfuscation of C_0 to an obfuscation of C_1 via the following sequence of hybrids.

- $\mathcal{H}_0$: This is the distribution $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(1^\lambda, C_0)$.
- $\mathcal{H}_1$: We modify the classical program $\mathsf{iO}_{\mathsf{class}}$ to always decode $\widetilde{c}$ to $c' = (0, C_0, 0^\ell)$ if it does not abort. This is accomplished by replacing the hardcoded (T_i, u_i) with (S_i, v_i) in $\mathsf{iO}_{\mathsf{class}}$ for each $i \in [2\ell+1]$. That is, we obfuscate the following program.

$\mathcal{H}_1$ Classical Program

Hard-Coded Values. Subspace and offset tuples $\{S_i, S_i, \mathsf{v}_i\}_{i\in[2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$.

Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

1: **if** $\mathsf{t}_i \in S_i + \mathsf{v}_i$ for every $i \in [2\ell+1]$ **then**
2: Let $\mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to, and define $c'_i := \widetilde{c}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
3: Parse $c'_1, \ldots, c'_{2\ell+1}$ as (b, C_0, C_1), where C_0 and C_1 are circuits with description length ℓ. Output $C_b(x)$.

Note that if $t_i \in S_i + v_i$, then $v'_i = v_i$. Since $\tilde{c}_i = c_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle$, the program always decodes $\widetilde{c}$ to $c' = c = (0, C_0, 0^\ell)$ if it does not abort.

- $\mathcal{H}_2$: Replace $c = (0, C_0, 0^\ell)$ with $c = (0, C_1, 0^\ell)$ when computing $\widetilde{c}$.
- $\mathcal{H}_3$: For each $i \in [2\ell+1]$, replace (S_i, v_i) with (T_i, u_i) in the obfuscated program P. This is the distribution $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C_1)$.

Indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ and between $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from subspace-hiding obfuscation (Corollary 4.10), while indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the security of $\mathsf{di}\mathcal{O}$ due to the fact that C_0 and C_1 differ on a polynomial number of inputs, which are hard to find. Recall that any $\mathsf{i}\mathcal{O}$ is a $\mathsf{di}\mathcal{O}$ for a polynomial number of differing inputs.

Certified everlasting security. Consider two functionally equivalent circuits C_0, C_1. We will switch an obfuscation of C_0 to an obfuscation of C_1, by changing one bit of the string c in the construction at a time, and argue that each switch is *statistically close* conditioned on the adversary producing a successful deletion certificate.

- $\mathcal{H}_0$: This is the certified everlasting security game with $|\widetilde{C}\rangle, \mathsf{vk} \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C_0)$.
- $\mathcal{H}_1 - \mathcal{H}_\ell$: In hybrid $\mathcal{H}_i$ for $i \in [1, \ldots, \ell]$, we switch the $\ell + 1 + i$'th bit of c to the i'th bit of the description of C_1. That is, in $\mathcal{H}_\ell$, the string $c = (0, C_0, C_1)$.
- $\mathcal{H}_{\ell+1}$: Switch the first bit of c to 1. So, now $c = (1, C_0, C_1)$.
- $\mathcal{H}_{\ell+2} - \mathcal{H}_{2\ell+1}$: In hybrid $\mathcal{H}_i$ for $i \in [\ell+2, \ldots, 2\ell+1]$, we switch the $i - \ell$'th bit of c to the $i - \ell - 1$'th bit of C_1. That is, in $\mathcal{H}_{2\ell+1}$, the string $c = (1, C_1, C_1)$.
- $\mathcal{H}_{2\ell+2}$: Switch the first bit of c to 0. So, now $c = (0, C_1, C_1)$.
- $\mathcal{H}_{2\ell+3} - \mathcal{H}_{3\ell+2}$: In hybrid $\mathcal{H}_i$ for $i \in [2\ell+3, 3\ell+2]$, we switch the $i - \ell - 1$'th bit of c to 0. That is, in $\mathcal{H}_{3\ell+2}$, the string $c = (0, C_1, 0^\ell)$. Note that this is exactly the certified everlasting security game with $|\widetilde{C}\rangle, \mathsf{vk} \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, C_1)$.

The proof follows by combining the following claims.

Claim 8.5. *For all* $i \in [1, \ldots, \ell]$, $\mathsf{TD}(\mathcal{H}_{i-1}, \mathcal{H}_i) = \mathrm{negl}(\lambda)$.

Proof. We will reduce this claim to Theorem 6.5. To do so, we must define a distribution $\mathcal{Z}$ and argue that it is *subspace-hiding* according to Definition 6.4. Defining $i^* := \ell + 1 + i$, we note that the output of di$\mathcal{O}$-CD in $\mathcal{H}_{i-1}$ and $\mathcal{H}_i$ can be written as

$$|(S_{i^*})_{\mathsf{v}_{i^*},\mathsf{w}_{i^*}}\rangle\,, \mathcal{Z}_\lambda(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$$

where a sample from $\mathcal{Z}_\lambda(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$ takes the form

$$\left(\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq i^*}, \mathsf{i}\mathcal{O}_{\mathsf{class}}, \left\{\mathsf{i}\mathcal{O}\left(P_{S_i^\perp+\mathsf{w}_i}\right)\right\}_{i\in[2\ell+1]}\right)$$

A sample from $\mathcal{Z}_\lambda(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$ is obtained by sampling from $\mathcal{H}_{i-1}$ (equivalently, $\mathcal{H}_i$) using fresh randomness for every $i \neq i^*$ and omitting the extra copy of $|(S_{i^*})_{\mathsf{v}_{i^*},\mathsf{w}_{i^*}}\rangle$. We show that $\mathcal{Z}_\lambda$ is subspace-hiding via the following sequence of hybrids.

- $\mathcal{H}_0(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$: This is $\mathcal{Z}_\lambda(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$.
- $\mathcal{H}_1(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$: Same as $\mathcal{H}_0$, except that we modify the classical program $\mathsf{i}\mathcal{O}_{\mathsf{class}}$ to always decode the first $\ell + 1$ bits of $\widetilde{c}$ to $(0, C_0)$ if it does not abort. Therefore it will always either abort or evaluate C_0.

  Specifically, replace the hard-coded values (T_i, u_i) with (S_i, v_i) in the di$\mathcal{O}$-CD classical program, for every $i \in [1, \ldots, \ell + 1]$. In other words, in $\mathcal{H}_1$, $\mathsf{i}\mathcal{O}_{\mathsf{class}}$ is an obfuscation of the following program.

  $\mathcal{H}_1$ Classical Program

  Hard-Coded Values. Subspace and offset tuples $\{S_i, S_i, \mathsf{v}_i\}_{i\in[\ell+1]}$ and $\{S_i, T_i, \mathsf{u}_i\}_{i\in[\ell+2,\ldots,2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$.
  Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

  1: **if** $\mathsf{t}_i \in S_i + \mathsf{v}_i$ for every $i \in [\ell + 1]$ **then**
  2: **if** $\mathsf{t}_i \in T_i + \mathsf{u}_i$ for every $i \in [\ell + 2, 2\ell + 1]$ **then**
  3: Let $\mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to, and define $c'_i := \widetilde{c}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
  4: Parse $c'_1, \ldots, c'_{2\ell+1}$ as (b, C_0, C_1), where C_0 and C_1 are circuits with description length ℓ.
  Output $C_b(x)$.

  Note that if $t_i \in S_i + v_i$, then $v'_i = v_i$. This is always the case for $i \in [\ell + 1]$ if the program does not abort. Since $\tilde{c}_i = c_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle$, the program always decodes the first $\ell + 1$ bits of $\widetilde{c}$ to $(0, C_0)$ if it does not abort.
- $\mathcal{H}_2(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$: Same as $\mathcal{H}_1$, except that we replace S_{i^*} with $0^{|S_{i^*}|}$ in the classical obfuscated program $\mathsf{i}\mathcal{O}_{class}$. This removes its dependence on S_{i*}. Note that since $i^* > \ell + 1$, the program still decodes the first ℓ bits of $\widetilde{c}$ to $(0, C_0)$ if it does not abort. Therefore it will always either abort or evaluate C_0.
- $\mathcal{H}_3(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$: Same as $\mathcal{H}_2$, except that we will remove the dependence of $\mathsf{i}\mathcal{O}\left(P_{S_{i^*}^\perp+\mathsf{w}_{i^*}}\right)$ on $S_{i^*} + \mathsf{w}_{i^*}$. Sample R_{i^*} as a uniformly random superspace of $S_{i^*}^\perp$ of dimension $3n/4$ and define $\mathsf{x}_{i^*} \in \mathsf{co}(R_{i^*})$ so that $S_{i^*}^\perp + \mathsf{w}_{i^*} \subset R_{i^*} + \mathsf{x}_{i^*}$. Use $\mathsf{i}\mathcal{O}\left(P_{R_{i^*}+\mathsf{x}_{i^*}}\right)$ in place of $\mathsf{i}\mathcal{O}\left(P_{S_{i^*}^\perp+\mathsf{w}_{i^*}}\right)$.

Note that this distribution can be prepared just given $(R_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{x}_{i^*}, \widetilde{c}_{i^*})$ as defined in Definition 6.4, and thus can be considered the simulated distribution.

Now, the indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_1$ follows by repeated application of subspace-hiding obfuscation (Corollary 4.10) for each $i \in [1, \ldots, \ell+1]$. Next, the indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the security of $\mathsf{i}\mathcal{O}$, since these programs are functionally equivalent. Indeed, note that in both hybrids, the program always outputs $C_0(x)$ if it does not abort, and the aborting conditions are the same. Finally, the indistinguishability of $\mathcal{H}_2$ and $\mathcal{H}_3$ follows again from Corollary 4.10.

□

Claim 8.6. $\mathsf{TD}(\mathcal{H}_\ell, \mathcal{H}_{\ell+1}) = \mathrm{negl}(\lambda)$.

Proof. We will again reduce this claim to Theorem 6.5. Note that the output of $\mathsf{di}\mathcal{O}$-CD in $\mathcal{H}_\ell$ and $\mathcal{H}_{\ell+1}$ can be written as

$$|(S_1)_{\mathsf{v}_1,\mathsf{w}_1}\rangle\,, \mathcal{Z}_\lambda(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$$

where a sample from $\mathcal{Z}_\lambda(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$ takes the form

$$\left(\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq 1}, \mathsf{i}\mathcal{O}_{\mathsf{class}}, \left\{\mathsf{i}\mathcal{O}\left(P_{S_i^\perp + \mathsf{w}_i}\right)\right\}_{i\in[2\ell+1]}\right)$$

A sample from $\mathcal{Z}_\lambda(S_{i^*}, T_{i^*}, \mathsf{u}_{i^*}, \mathsf{w}_{i^*}, \widetilde{c}_{i^*})$ is obtained by sampling from $\mathcal{H}_\ell$ (equivalently, $\mathcal{H}_{\ell+1}$) using fresh randomness for every $i \neq 1$ and omitting the extra copy of $|(S_1)_{\mathsf{v}_1,\mathsf{w}_1}\rangle$. In $\mathcal{H}_\ell$, we have $\widetilde{c}_1 = 0 \oplus \langle \mathsf{v}_1, \mathbf{1}\rangle$ and in $\mathcal{H}_{\ell+1}$, we have $\widetilde{c}_1 = 1 \oplus \langle \mathsf{v}_1, \mathbf{1}\rangle$. We show that $\mathcal{Z}_\lambda$ is subspace-hiding via the following sequence of hybrids.

- $\mathcal{H}_0(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$: This is $\mathcal{Z}_\lambda(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$.
- $\mathcal{H}_1(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$: Same as $\mathcal{H}_0$, except that we modify the classical obfuscated program $\mathsf{i}\mathcal{O}_{\mathsf{class}}$ to always decode the last 2ℓ bits of $\widetilde{c}$ to (C_0, C_1). Therefore it will always abort, output $C_0(x)$, or output $C_1(x)$, depending on $\widetilde{c}_1$. Specifically, replace (T_i, u_i) with (S_i, v_i) in the $\mathsf{di}\mathcal{O}$-CD classical program for every $i \in [2, \ldots, 2\ell+1]$.

  Note that if $t_i \in S_i + v_i$, then $v_i' = v_i$. This is always the case for $i > 1$ if the program does not abort. Since $\tilde{c}_i = c_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle$, the program always decodes the last $2\ell+$ bits of $\widetilde{c}$ to (C_0, C_1) if it does not abort.
- $\mathcal{H}_2(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$: Same as $\mathcal{H}_1$, except we modify the classical program to use hard-coded versions of C_0 and C_1, instead of decoding them from $\widetilde{c}$. In particular, it uses the indistinguishability obfuscations $\widetilde{C}_0$ and $\widetilde{C}_1$ of these programs. In detail, $\mathsf{i}\mathcal{O}_{\mathsf{class}}$ is an obfuscation of the following program:

$\mathcal{H}_2$ Classical Program

Hard-Coded Values. Subspace and offset tuple $\{S_1, T_1, \mathsf{u}_1\}$ and $\{S_i, S_i, \mathsf{v}_i\}_{i\in[2,\ldots,2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$, obfuscated programs $\widetilde{C}_0 = \mathsf{i}\mathcal{O}(C_0)$ and $\widetilde{C}_1 = \mathsf{i}\mathcal{O}(C_1)$.
Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

1: **if** $\mathsf{t}_1 \in T_1 + \mathsf{u}_1$ **then**
2: **if** $\mathsf{t}_i \in S_i + \mathsf{v}_i$ for every $i \in [1, \ldots, 2\ell+1]$ **then**

3: Let $\mathsf{v}'_1 \in \mathsf{co}(S_1)$ be the coset of S_1 that t_1 belongs to, and define $b := \widetilde{c}_1 \oplus \langle \mathsf{v}'_1, \mathbf{1} \rangle$.
4: Output $\widetilde{C}_b(x)$.

- $\mathcal{H}_3(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$: Same as $\mathcal{H}_2$, except in the classical program, the hard-coded value $\widetilde{C}_0$ is an obfuscation of C_1 instead of C_0. In other words, $\mathsf{i}\mathcal{O}_{class}$ is an obfuscation of the following program:

$\mathcal{H}_3$ Classical Program

Hard-Coded Values. Subspace and offset tuple $\{S_1, T_1, \mathsf{u}_1\}$ and $\{S_i, S_i, \mathsf{v}_i\}_{i\in[2,\ldots,2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$, obfuscated programs $\widetilde{C}_0 = \mathsf{i}\mathcal{O}(C_1)$ and $\widetilde{C}_1 = \mathsf{i}\mathcal{O}(C_1)$.
Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

1: **if** $\mathsf{t}_1 \in T_1 + \mathsf{u}_1$ **then**
2: **if** $\mathsf{t}_i \in S_i + \mathsf{v}_i$ for every $i \in [1, \ldots, 2\ell+1]$ **then**
3: Let $\mathsf{v}'_1 \in \mathsf{co}(S_1)$ be the coset of S_1 that t_1 belongs to, and define $b := \widetilde{c}_1 \oplus \langle \mathsf{v}'_1, \mathbf{1} \rangle$.
4: Output $\widetilde{C}_b(x)$.

- $\mathcal{H}_4(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$: Same as $\mathcal{H}_3$, except the classical program always evaluates a hard-coded copy of C_1 if it does not abort. Since the circuit being internally evaluated is always the same, the program does not need to decode $\widetilde{c}$, and therefore we can also remove S_1 from its parameters. In detail, $\mathsf{i}\mathcal{O}_{\mathsf{class}}$ is an obfuscation of the following program:

$\mathcal{H}_4$ Classical Program

Hard-Coded Values. A subspace and offset $\{T_1, \mathsf{u}_1\}$, subspace and offset tuples $\{S_i, S_i, \mathsf{v}_i\}_{i\in[2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$, the circuit C_1.
Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

1: **if** $\mathsf{t}_1 \in T_1 + \mathsf{u}_1$ **then**
2: **if** $\mathsf{t}_i \in S_i + \mathsf{v}_i$ for every $i \in [1, \ldots, 2\ell+1]$ **then**
3: Output $C_1(x)$.

- $\mathcal{H}_5(S_1, T_1, \mathsf{u}_1, \mathsf{w}_1, \widetilde{c}_1)$: Same as $\mathcal{H}_4$, except that we will remove the dependence of $\mathsf{i}\mathcal{O}\left(P_{S_1^\perp + \mathsf{w}_1}\right)$ on $S_1 + \mathsf{w}_1$. Sample R_1 as a uniformly random superspace of $S_1^\perp$ of dimension $3n/4$ and define $\mathsf{x}_1 \in \mathsf{co}(R_1)$ so that $S_1^\perp + \mathsf{w}_1 \subset R_1 + \mathsf{x}_1$. Use $\mathsf{i}\mathcal{O}\left(P_{R_1+\mathsf{x}_1}\right)$ in place of $\mathsf{i}\mathcal{O}\left(P_{S_1^\perp+\mathsf{w}_1}\right)$. Note that this distribution can be prepared just given $(R_1, T_1, \mathsf{u}_1, \mathsf{x}_1, \widetilde{c}_1)$ as defined in Definition 6.4, and thus can be considered the simulated distribution.

Now, the indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_1$ follows by repeated application of Corollary 4.10 for each $i \in [2, \ldots, 2\ell+1]$.

Next, the indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the security of $\mathsf{i}\mathcal{O}$, since these programs are functionally equivalent. Indeed, note that in $\mathcal{H}_1$, the program will always either abort or unmask the $\widetilde{c}$ as (b, C_0, C_1) for some arbitrary bit b. Therefore in both $\mathcal{H}_1$ and $\mathcal{H}_2$, the program outputs $C_b(x)$ if it does not abort.

The indistinguishability of $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from the security of di$\mathcal{O}$, which follows from the security of i$\mathcal{O}$. Since C_0 and C_1 differ on a polynomial number of hard-to-find inputs, i$\mathcal{O}(C_0) \approx$ i$\mathcal{O}(C_1)$. In more detail, assuming $\mathcal{H}_2$ and $\mathcal{H}_3$ are distinguishable, we can distinguish i$\mathcal{O}(C_0)$ from i$\mathcal{O}(C_1)$ as follows. The reduction is given (C_0, C_1, aux) sampled by the differing inputs circuits sampler, as well as i$\mathcal{O}(C_b)$ for a random b. It classically samples the cosets, then constructs the obfuscated program specified by $\mathcal{H}_2/\mathcal{H}_3$ using i$\mathcal{O}(C_b)$. It runs the distinguisher for $\mathcal{H}_2/\mathcal{H}_3$ and outputs the result.

The indistinguishability of $\mathcal{H}_3$ and $\mathcal{H}_4$ follows from the security of i$\mathcal{O}$, since these programs are functionally equivalent. Indeed, note that in $\mathcal{H}_3$, the program will always evaluate i$\mathcal{O}(C_1)$ if it does not abort, which is functionally equivalent to C_1. Finally, the indistinguishability of $\mathcal{H}_4$ and $\mathcal{H}_5$ follows again from Corollary 4.10.

□

Claim 8.7. *For all* $i \in [\ell + 2, \ldots, 2\ell + 1]$, $\mathsf{TD}(\mathcal{H}_{i-1}, \mathcal{H}_i) = \mathrm{negl}(\lambda)$.

Proof. This follows from essentially an identical proof as Claim 8.5. □

Claim 8.8. $\mathsf{TD}(\mathcal{H}_{2\ell+1}, \mathcal{H}_{2\ell+2}) = \mathrm{negl}(\lambda)$.

Proof. This follows from a similar proof to Claim 8.6. The only difference is that we can transition directly from sub-hybrid $\mathcal{H}_1$ to $\mathcal{H}_4$ using the security of i$\mathcal{O}$. Note that in $\mathcal{H}_1$, the classical program would always unmask (b, C_1, C_1) if it does not abort, and so it always evaluates C_1 in this case. In $\mathcal{H}_4$, the classical program also always evaluates C_1 if it does not abort, and the aborting conditions are the same. □

Claim 8.9. *For all* $i \in [2\ell + 3, \ldots, 3\ell + 2]$, $\mathsf{TD}(\mathcal{H}_{i-1}, \mathcal{H}_i) = \mathrm{negl}(\lambda)$.

Proof. This follows from essentially an identical proof as Claim 8.5. □

□

8.3 Extensions

We describe several extensions to di$\mathcal{O}$-CD. These include:

- Section 8.3.1: Nested di$\mathcal{O}$-CD.
- Section 8.3.2: di$\mathcal{O}$-CD with provable correctness.
- Section 8.3.3: Strong secure software leasing.
- Section 8.3.4: Succinct obfuscation and two-message blind delegation with certified deletion.
- Section 8.3.5: Oracles with certified deletion.

8.3.1 Nested Differing Inputs Obfuscation with Certified Deletion

Recall that the di$\mathcal{O}$-CD construction given above can be evaluated by performing a computational basis measurement, then evaluating a classical program on the result. Therefore, we can "nest" an obfuscated program $((|\psi\rangle, \mathsf{i}\mathcal{O}_{\mathsf{class}}), \mathsf{vk}) \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(C)$ inside another program C_f by creating the program $C_f \circ \mathsf{i}\mathcal{O}_{\mathsf{class}}$. It can be correctly evaluated by measuring $|\psi\rangle$ in the computational basis, then evaluating $C_f \circ \mathsf{i}\mathcal{O}_{\mathsf{class}}$ on the measurement result and any other inputs. Furthermore, the inner program retains its certified deletion security. $C_f \circ \mathsf{i}\mathcal{O}_{\mathsf{class}}$ can be additionally obfuscated using di$\mathcal{O}$-CD to protect C_f, since $C_f \circ \mathsf{i}\mathcal{O}_{\mathsf{class}}$ is a classical program. Thus we have the following corollary.

Corollary 8.10. *Assuming post-quantum indistinguishability obfuscation and one-way functions, there exists nested differing inputs obfuscation with (publicly verifiable) certified deletion for polynomially many nested differing inputs.*

8.3.2 di$\mathcal{O}$-CD with Provable Correctness

A desirable property for an obfuscation scheme is the ability to prove that the program is well-formed. For example, when obfuscating a program that allows the holder to decrypt ciphertexts, the holder may wish to be assured that they can correctly decrypt any ciphertext. Unfortunately, our di$\mathcal{O}$-CD construction does not allow for this. Since the functionality of an obfuscated program $\widetilde{C} = (|\psi\rangle, \mathsf{i}\mathcal{O}_{\mathsf{class}})$ is determined by $|\psi\rangle$, whether or not $\widetilde{C}$ has the correct functionality is not a QMA statement.

Tokenized di$\mathcal{O}$-CD To remedy this, we introduce a new di$\mathcal{O}$-CD property which we call "tokenization" and give an alternative scheme which satisfies it. A tokenized di$\mathcal{O}$-CD generates programs which consist of a quantum token and a classical (obfuscated) program. Crucially, the functionality of the program is fully determined by the classical obfuscated program. On input a (measured) token t and an input x, the classical program either aborts for all x or evaluates $C(x)$ for a fixed program C. This ensures that if a token is valid, then the program behaves correctly on it, and that invalid tokens are detectable. Thus, the well-formedness of a tokenized di$\mathcal{O}$-CD program can be formulated as the QMA statement "there exists a token t such that the classical program evaluates $C(x)$ for all x".

Definition 8.11 (Tokenized di$\mathcal{O}$-CD)**.** *A (nested) differing inputs obfuscator with certified deletion* (Obf, Eval, Del, Verify) *is* tokenized *if it satisfies the following properties:*

1. $\mathsf{Obf}(\Pi)$ *outputs a quantum token* $|\psi\rangle = \sum_{t\in\{0,1\}^{\mathrm{poly}(\lambda)}} \alpha_t |t\rangle$ *and a classical program* $\widetilde{\Pi}$*.*
2. $\mathsf{Eval}((|\psi\rangle, \widetilde{\Pi}), x) = \widetilde{\Pi}(|\psi\rangle, x) = \sum_{t\in\{0,1\}^{\mathrm{poly}(\lambda)}} \alpha_t |\widetilde{\Pi}(t, x)\rangle$.
3. *The probability of generating an obfuscation* $(|\psi\rangle, \widetilde{\Pi}) \leftarrow \mathsf{Obf}(\Pi)$ *such that for all* $t \in \{0,1\}^{\mathrm{poly}(\lambda)}$*, either*
   (a) for all x*,* $\widetilde{\Pi}(t, x) = \Pi(x)$
   (b) or for all x*,* $\widetilde{\Pi}(t, x) = \bot$

   is $1 - \mathrm{negl}(\lambda)$*.*

Computational Certified Deletion Achieving the tokenized property requires trading statistical security after deletion for computational security after deletion. This is unavoidable for provable correctness, since the functionality must be encoded classically in order for correctness to be a QMA statement. Thus the functionality is information-theoretically determined even after deletion. We define a computational certified deletion security by directly leaking the differing inputs after deletion. This captures the fact that the adversary can no longer evaluate the program on differing inputs once it deletes the program.

Definition 8.12 (Computational certified deletion for di$\mathcal{O}$-CD)**.** *Let $\mathcal{D}$ be the distribution associated with a differing inputs circuit family $\mathcal{C}$. Define the following game, parameterized by a bit b:*

1. *The challenger samples differing input circuits $(C_0,\ C_1,\ \mathsf{aux}) \leftarrow \mathcal{D}$. It computes the set of differing inputs $Y^* = \{y^* : C_0(y^*) \neq C_1(y^*)\}$ and the obfuscation $(\widetilde{C_b}, \mathsf{vk}) \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(\Pi_b, 1^\lambda)$, then sends $(\widetilde{C}_b, C_0, C_1, \mathsf{aux})$ to the adversary.*
2. *The challenger receives a proof of deletion cert from the adversary.*
3. *If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top$, send the set of differing inputs Y^* to the adversary.*
4. *The adversary outputs a bit b' and wins if $b' = b$.*

A $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}$ scheme has computational certified deletion security for $\mathcal{C}$ if the adversary's advantage in winning this game is $\mathrm{negl}(\lambda)$.

If this holds even when the adversary also receives vk in step 1, then we say the $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}$ scheme has publicly verifiable computational certified deletion security.

Construction The construction is almost exactly the same as our original di$\mathcal{O}$-CD construction (Section 8.2), except we remove the noise from the check. This is accomplished by setting $S_i = T_i$ and $\mathsf{u}_i = \mathsf{v}_i$ instead of sampling them so that $T_i + \mathsf{u}_i$ is a random super-coset of $S_i + \mathsf{v}_i$. In more detail, to obfuscate a program C, do the following.

- Let $n = 4\lambda$, let $\ell = |C|$. For each $i \in [2\ell + 1]$, sample $S_i < \mathbb{F}_2^n$ such that $\dim(S_i) = n/2$ and sample $\mathsf{v}_i, \mathsf{w}_i \leftarrow \mathsf{co}(S_i) \times \mathsf{co}(S_i^\perp)$.
- Let $c := (0, C, 0^\ell)$, and for all $i \in [2\ell + 1]$, define $\widetilde{c}_i := c_i \oplus \langle \mathsf{v}_i, \mathbf{1} \rangle$. Define $\widetilde{c} := (\widetilde{c}_1, \ldots, \widetilde{c}_{2\ell+1})$.
- Compute the indistinguishability obfuscation $\mathsf{i}\mathcal{O}_{\mathsf{class}}$ of the $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}$ classical program using hard-coded values $\{S_i, S_i, \mathsf{v}_i\}_{i\in[2\ell+1]}$ and $\widetilde{c}$.
- Output

$$|\widetilde{C}\rangle := \left(\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]},\ \mathsf{i}\mathcal{O}_{\mathsf{class}}\right), \quad \mathsf{vk} := \left\{\mathsf{i}\mathcal{O}\left(P_{S_i^\perp + \mathsf{w}_i}\right)\right\}_{i\in[2\ell+1]}.$$

Deletion and verification are done as in the original scheme. For convenience, we recall the $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}$ classical program here.

diO-CD Classical Program

Hard-Coded Values. Subspace and offset tuples $\{S_i, T_i, \mathsf{u}_i\}_{i\in[2\ell+1]}$, a bitstring $\widetilde{c} \in \{0,1\}^{2\ell+1}$.

Input. vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ and an input x.

1: **if** $\mathsf{t}_i \in T_i + \mathsf{u}_i$ for every $i \in [2\ell+1]$ **then**
2: Let $\mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to, and define $c'_i := \widetilde{c}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
3: Parse $c'_1, \ldots, c'_{2\ell+1}$ as (b, C_0, C_1), where C_0 and C_1 are circuits with description length ℓ. Output $C_b(x)$.

Corollary 8.13 (Tokenized diO-CD)**.** *Assuming post-quantum indistinguishability obfuscation and one-way functions, there exists* diO-CD with provable correctness *for polynomially many differing inputs.*

Proof. Observe that if $\mathsf{t}_i \in S_i + \mathsf{v}_i$, then $\widetilde{c}$ is always unmasked to $(0, C, 0^\ell)$. Therefore whenever the program does not abort, it evaluates C. Since whether the program aborts is independent of the input x, the above construction is indeed tokenized. This establishes provable correctness.

Next we show (publicly verifiable) computational certified deletion security. Consider the following hybrids:

- $\mathcal{H}_0$: The outcome bit from the computational certified deletion game for the tokenized diO-CD construction.
- $\mathcal{H}_1$: This is the same as $\mathcal{H}_0$, except we modify $\mathsf{iO}_{\mathsf{class}}$. For every $i \in [2\ell+1]$, sample a random $T_i \supset S_i$ with dimension $3n/4$. Let u_i be the coset of T_i that v_i belongs to. Instead of obfuscating the diO-CD classical program with $\{S_i, S_i, \mathsf{v}_i\}$ hard-coded, use $\{S_i, T_i, \mathsf{u}_i\}$. This is our original diO-CD scheme.

Indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_1$ follows from the repeated application of subspace-hiding obfuscation (Corollary 4.10). Observe that any QPT adversary's winning advantage in $\mathcal{H}_1$ is negligible. Otherwise, an adversary could violate the (information-theoretic) certified deletion security of the original diO-CD scheme (Theorem 8.4) by computing the list of differing inputs inefficiently after deletion, then running the QPT adversary. Since the distributions over the outcome bits of the game in $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable, this also holds in $\mathcal{H}_0$.

□

8.3.3 Strong Secure Software Leasing

We show that obfuscation with certified deletion implies a strong notion of secure software leasing [AL21] for a wide class of programs. As corollary, we construct this notion of secure software leasing from post-quantum indistinguishability obfuscation and post-quantum one-way functions.

Definition 8.14 (Secure Software Leasing)**.** *A secure software leasing scheme for a circuit class $\mathcal{C}$ consists of the QPT algorithms* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$*, defined as follows.*

- $\mathsf{Gen}(1^n, C)$ *takes in the security parameter and a circuit $C \in \mathcal{C}$, then outputs a leased program $\widetilde{C}$ and a verification key* vk*.*
- $\mathsf{Eval}(\widetilde{C}, x)$ *takes in a leased program $\widetilde{C}$ and an input x, then outputs a value y.*

- $\mathsf{Verify}(\mathsf{vk}, \widetilde{C})$ *takes in a verification key* vk *and a leased program* $\widetilde{C}$*, then outputs Accept or Reject.*

It must satisfy correctness:

- ***Evaluation Correctness:*** *For every* $C \in \mathcal{C}$ *with input length* n*,*

$$\Pr[\mathsf{Eval}(\widetilde{C}, x) = C(x)\ \forall x \in \{0,1\}^n : (\widetilde{C}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n, C)] = 1 - \mathrm{negl}(\lambda)$$

- ***Verification Correctness:*** *For every* $C \in \mathcal{C}$ *with input length* n*,*

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \widetilde{C}) = \top : (\widetilde{C}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n, C)] = 1 - \mathrm{negl}(\lambda)$$

The definition above has slightly different syntax than the one presented in [AL21]. It directly generates a leased program and verification key in Gen, instead of generating the verification key in Gen, then separately generating leased programs in an algorithm named Lessor which also takes in the verification key. We note that a scheme with the syntax definition above can be transformed into a scheme with the syntax from [AL21] by using any symmetric-key encryption scheme and signature scheme.

Strong finite-term leasing security guarantees that if the leasee returns a valid program, then the output of the program on certain inputs is hidden. This is a stronger guarantee than finite lessor security, which only guarantees that after the leasee returns a valid program, they cannot evaluate every input using the honest Eval procedure.

Definition 8.15 (Strong Finite-Term Leasing)**.** *A secure software leasing scheme* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Verify})$ *for a circuit class* $\mathcal{C}$ *associated with a distribution* $\mathcal{D}_{\mathcal{C}}$ *has strong* β*-perfect finite-term leasing if for all QPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *where* $\mathcal{A}_1$ *outputs a bipartite state on registers* R_1 *and* R_2*, the following holds:*

$$\Pr\left[\begin{array}{cc} \mathsf{Verify}(\mathsf{vk}, \mathsf{Tr}^{R_2}[\rho]) = \top & C \leftarrow \mathcal{D}_{\mathcal{C}} \\ \wedge & : \ (\widetilde{C}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n, C) \\ \forall x\ \Pr[\mathcal{A}_2(\mathsf{Tr}^{R_1}[\rho], x) = C(x)] \geq \beta & \rho \leftarrow \mathcal{A}_1(\widetilde{C}) \end{array}\right] = \mathrm{negl}(\lambda)$$

If this holds when vk *is also given to* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*, then we say it has publicly verifiability. If this holds when* $\mathcal{A}_2$ *is computationally unbounded, we say it has statistical returns.*

We note that unlike in di$\mathcal{O}$, $\mathcal{A}$ does not receive additional auxiliary input after the circuit C is sampled.

Theorem 8.16. *Assuming* di$\mathcal{O}$-CD*, there exists secure software leasing with (publicly verifiable)* β*-perfect strong finite-term security for all differing inputs circuits families, where* $\beta = 1/2 + \mathrm{negl}(\lambda)$*. Furthermore, it has statistical returns.*

Proof. Let di$\mathcal{O}$-CD $= (\mathsf{Obf}, \mathsf{Eval}, \mathsf{Del}, \mathsf{Verify})$ be a differing inputs obfuscation with certified deletion. The secure software leasing scheme SSL is

- $\mathsf{SSL.Gen}(1^n, C)$: Run di$\mathcal{O}$-CD.$\mathsf{Obf}(1^n, C)$ then output the result
- $\mathsf{SSL.Eval}(\widetilde{C}, x)$: Run di$\mathcal{O}$-CD.$\mathsf{Eval}(\widetilde{C}, x)$ and outputs the result.
- $\mathsf{SSL.Verify}(\mathsf{vk}, \widetilde{C})$: We first describe the scheme with measurements. Evaluate $\mathsf{cert} \leftarrow$ di$\mathcal{O}$-CD.$\mathsf{Del}(\widetilde{C})$. Output di$\mathcal{O}$-CD.$\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$. To avoid damaging a valid program, do this procedure coherently and measure the output bit.

Correctness follows from the description of the scheme and correctness of $\mathsf{di}\mathcal{O}$-CD. To show strong finite term lessor security, we first define $\mathcal{D}_{\mathcal{C}}$. Recall that a differing inputs circuit family is associated with an efficiently sampleable distribution $\mathcal{D}_{\mathsf{DI}}$. To generate a sample from $\mathcal{D}_{\mathcal{C}}$, sample $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_{\mathsf{DI}}$ and a bit b, then output C_b. Observe that if the leasee returns a valid program, then a valid deletion certificate can be extracted from the program.

Consider the SSL experiment where $\mathcal{D}_{\mathcal{C}}$ outputs a circuit C_b. Call the state that the adversary outputs $\rho(b)$.[18] Let $\rho'(b)$ be the leftover state after applying the deletion procedure to the first register and tracing out the resulting certificate. Say the certificate is valid with noticeable probability. Then by the security of $\mathsf{di}\mathcal{O}$-CD and convexity of trace distance, we have $\mathsf{TD}(\rho'(0), \rho'(1)) = \mathrm{negl}(\lambda)$ whenever the certificate is valid.

We now argue that the probability $\rho'(b)$ can be used to evaluate $C_b(y^*)$ is at most $1/2 + \mathrm{negl}(\lambda)$ for any differing input y^*. Say that $\mathcal{A}_2(\rho'(b), x)$ outputs $C_b(x)$ with probability β_0 for all x. Then $\mathcal{A}_2(\rho'(1-b), x)$ also outputs $C_b(x)$ with probability $\geq \beta_0 - \mathrm{negl}(\lambda)$. Therefore, for any differing input y^*, $\mathcal{A}_2(\rho'(1-b), y^*)$ is incorrect with probability at least $1 - \beta_0 - \mathrm{negl}(\lambda)$. Thus, the probability of $\mathcal{A}_2$ correctly evaluating for a random b is at most $1/2\beta_0 + 1/2(1 - (\beta_0 - \mathrm{negl})) = 1/2 + \mathrm{negl}(\lambda)$. In other words, $\beta \leq 1/2 + \mathrm{negl}(\lambda)$. □

Reducing β. We can lower β further for certain differing inputs circuits classes. Consider the game where an adversary $\mathcal{A}$ receives $\widetilde{C_0} \leftarrow \mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(C_0)$, deletes it, then attempts to guess $C_0(y^*)$ for some differing input y^*. Since $\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}(C_0) \approx \mathsf{di}\mathcal{O}(C_1)$ even given y^* after deletion, intuitively $\mathcal{A}$ cannot guess $C_0(y^*)$ any better than if it were just given C_1 and y^*.

Theorem 8.17. *Assuming* $\mathsf{di}\mathcal{O}$-CD, *there exists secure software leasing with (publicly verifiable) β-perfect strong finite-term security for all differing inputs circuits families, where*

$$\beta = \max_{QPT\ \mathcal{A}} \Pr\left[\mathcal{A}(C_1, y^*) = C_0(y^*) : \begin{array}{c}(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_{\mathcal{C}} \\ \textit{uniform } y^* \textit{ s.t. } C_0(y^*) \neq C_1(y^*)\end{array}\right] + \mathrm{negl}(\lambda)$$

Furthermore, it has statistical returns, where β is taken as the maximum over unbounded $\mathcal{A}$.

Proof. The proof is almost identical to the one above, except for two differences. First, we define $\mathcal{D}_{\mathcal{C}}$ to output a random C_0, instead of a random C_b. Second, we note that $\mathcal{A}_2(\rho'(1), y^*)$ outputs $C_0(y^*)$ with probability at most β for every differing input y^*. Therefore $\mathcal{A}_2(\rho'(0), y^*)$ outputs $C_0(y^*)$ with probability at most $\beta + \mathrm{negl}(\lambda)$. □

This result gives a very general criteria for whether a program class can be securely leased. Indeed, many program classes which were previously studied for secure software leasing are a special case of this theorem. We do note, however, that some of these classes have been securely leased in prior work using weaker assumptions than $\mathsf{i}\mathcal{O}$.

Corollary 8.18. *Assuming post-quantum one-way functions and* $\mathsf{di}\mathcal{O}$-CD, *there exists strong secure software leasing for pseudorandom functions, evasive functions, random point functions, and compute-and-compare circuits.*

[18] We consider this to be a pure state sampled from the adversary's output distribution. Otherwise, we can set the probability of outputting a "good" pirated state to 0 by simply arguing that the adversary *always* outputs mixed state with negligible probability mass on "good" pure states.

Sketch. We sketch the result for psuedorandom functions and note that the other classes can be argued similarly. Let $\mathcal{C}$ be a PRF-evaluating circuit class, where $C_k(x) = \mathsf{PRF}(k, x)$ for every $C_k \in \mathcal{C}$. Let k_{y^*} be a privately punctured PRF key [KPTZ13, BW13, BGI14, BLW17], which has the same behavior as k, except it contains no information about $\mathsf{PRF}(k, y^*)$. Privately puncturable PRFs can be obtained from $\mathsf{i}\mathcal{O}$ [BLW17], which is implied by $\mathsf{di}\mathcal{O}$-CD. Then $\{(C_k, C_{k_y^*}) : y^* \leftarrow \{0,1\}^\lambda\}$ is a differing inputs circuits class where

$$\max_{\text{QPT } \mathcal{A}} \Pr\left[\mathcal{A}(C_1, y^*) = C_0(y^*) : \begin{array}{c}(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_{\mathcal{C}} \\ \text{uniform } y^* \text{ s.t. } C_0(y^*) \neq C_1(y^*)\end{array}\right] = \mathrm{negl}(\lambda)$$

□

8.3.4 Succinct obfuscation and two-message blind delegation with certified deletion

Now, we observe that exactly the same proof given in Theorem 8.4, but using $\mathsf{si}\mathcal{O}$ rather than $\mathsf{i}\mathcal{O}$, will suffice to prove the following theorem.

Theorem 8.19. *Assuming post-quantum succinct indistinguishability obfuscation and one-way functions, there exists succinct indistinguishability obfuscation with (publicly-verifiable) certified deletion (Definition 8.3).*

We will use this observation to construct a *two-message* blind delegation with certified deletion protocol. In fact, we will present our construction using the primitive of *succinct randomized encoding with certified deletion*, which is defined as follows.

Definition 8.20 (Succinct randomized encoding with certified deletion)**.** *A succinct randomized encoding with (publicly-verifiable) certified deletion for a class of Turing machines $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ has the following syntax.*

- $\mathsf{SRE\text{-}CD.Enc}(1^\lambda, M, x, t) \to |\widetilde{M}_{x,t}\rangle, \mathsf{vk}$*: The encode algorithm takes as input the security parameter 1^λ, the description of a Turing machine $M \in \mathcal{M}_\lambda$, an input x, and a step-size t, and outputs a (quantum) succinct randomized encoding $|\widetilde{M}_{x,t}\rangle$ and a verification key* vk*.*
- $\mathsf{SRE\text{-}CD.Eval}(|\widetilde{M}_{x,t}\rangle) \to y$*: The evaluation algorithm takes as input the succinct randomized encoding $|\widetilde{M}_{x,t}\rangle$ and outputs a (classical) y.*
- $\mathsf{SRE\text{-}CD.Del}(|\widetilde{M}_{x,t}\rangle) \to \mathsf{cert}$*: The deletion algorithm takes as input the obfuscated program $|\widetilde{M}_{x,t}\rangle$ and outputs a deletion certificate* cert*.*
- $\mathsf{SRE\text{-}CD.Verify}(\mathsf{vk}, \mathsf{cert}) \to \{\top, \bot\}$*: The verification algorithm takes as input the verification key and a deletion certificate and outputs either $\top$ or $\bot$.*

It should satisfy the following properties.

- ***Correctness.*** *For all $\lambda \in \mathbb{N}$, all $M \in \mathcal{M}_\lambda$, all $x \in \{0,1\}^n$, and all t,*

$$\Pr[\mathsf{SRE\text{-}CD.Eval}(|\widetilde{M}_{x,t}\rangle) = M^t(x) : |\widetilde{M}_{t,x}\rangle, \mathsf{vk} \leftarrow \mathsf{SRE\text{-}CD}(1^\lambda, M, x, t)] = 1.$$

- ***Correctness of deletion.*** *For all sequences of Turing machines and inputs* $\{M_\lambda \in \mathcal{M}_\lambda, x_\lambda, t_\lambda\}_{\lambda\in\mathbb{N}}$*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk},\mathsf{cert}) = \top : \begin{array}{r} |\widetilde{M}_{x,t}\rangle\,, \mathsf{vk} \leftarrow \mathsf{SRE\text{-}CD}(1^\lambda, M_\lambda, x_\lambda, t_\lambda) \\ \mathsf{cert} \leftarrow \mathsf{SRE\text{-}CD.Del}(|\widetilde{M}_{x,t}\rangle) \end{array}\right] = 1 - \mathrm{negl}(\lambda).$$

- ***Computational security.*** *There exists a QPT simulator* $\{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ *such that for all sequences of Turing machines and inputs* $\{M_\lambda \in \mathcal{M}_\lambda, x_\lambda, t_\lambda\}_{\lambda\in\mathbb{N}}$ *and all QPT adversaries* $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$*,*

$$\left|\Pr\left[\mathcal{A}_\lambda\left(\mathsf{SRE\text{-}CD}(1^\lambda, M_\lambda, x_\lambda, t_\lambda)\right) = 1\right] - \Pr\left[\mathcal{A}_\lambda\left(\mathcal{S}_\lambda(1^\lambda, M_\lambda, t_\lambda, M_\lambda^{t_\lambda}(x_\lambda))\right) = 1\right]\right| = \mathrm{negl}(\lambda).$$

- ***Certified everlasting security.*** *There exists a QPT simulator* $\{\mathsf{S}_\lambda\}_{\lambda\in\mathbb{N}}$ *such that for all sequences of Turing machines and inputs* $\{M_\lambda \in \mathcal{M}_\lambda, x_\lambda, t_\lambda\}_{\lambda\in\mathbb{N}}$ *and all QPT adversaries* $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$*,*

$$\mathsf{TD}\left(\mathsf{REAL}^{\mathcal{A}_\lambda}(M_\lambda, x_\lambda, t_\lambda), \mathsf{IDEAL}^{\mathcal{A}_\lambda,\mathcal{S}_\lambda}(M_\lambda, t_\lambda, M_\lambda^{t_\lambda}(x_\lambda))\right) = \mathrm{negl}(\lambda),$$

where the experiments are defined as follows. Given a Turing machine / input (M, x, t) *and an adversary* $\mathcal{A}$*, the experiment* $\mathsf{REAL}^{\mathcal{A}}(M, x, t)$ *is defined as follows.*

  - *Sample* $|\widetilde{M}_{x,t}\rangle\,, \mathsf{vk} \leftarrow \mathsf{SRE\text{-}CD}(1^\lambda, M, x, t)$ *and initialize* $\mathcal{A}$ *with* $(|\widetilde{M}_{x,t}\rangle\,, \mathsf{vk})$*.*
  - *Parse* $\mathcal{A}$*'s output as a deletion certificate* cert *and a left-over quantum state* ρ*.*
  - *If* $\mathsf{SRE\text{-}CD.Verify}(\mathsf{vk}, \mathsf{cert}) = \top$ *then output* ρ*, and otherwise output* $\bot$*.*

  Given a Turing machine / output (M, t, y)*, an adversary* $\mathcal{A}$*, and a simulator* $\mathcal{S}$*, the experiment* $\mathsf{IDEAL}^{\mathcal{A},\mathcal{S}}(M, t, y)$ *is defined as follows.*

  - *Sample* $|\widetilde{M}_y\rangle\,, \mathsf{vk} \leftarrow \mathcal{S}(1^\lambda, M, t, y)$ *and initialize* $\mathcal{A}$ *with* $(|\widetilde{M}_y\rangle\,, \mathsf{vk})$*.*
  - *Parse* $\mathcal{A}$*'s output as a deletion certificate* cert *and a left-over quantum state* ρ*.*
  - *If* $\mathsf{SRE\text{-}CD.Verify}(\mathsf{vk}, \mathsf{cert}) = \top$ *then output* ρ*, and otherwise output* $\bot$*.*

- ***Succinctness.*** *The running time of* $\mathsf{SRE\text{-}CD}$ *must be* $\mathrm{poly}(\lambda, |M|, |x|, \log t)$*.*

Note that $\mathsf{SRE\text{-}CD}$ is immediately implied by an si$\mathcal{O}$-CD scheme, by obfuscating an input-less Turing machine $M[x]$ that has x hard-coded. The simulator will simply obfuscate an input-less Turing machine that always outputs $M^t(x)$.

Now, we show how to construct a two-message blind delegation protocol with certified deletion. We will define an ideal functionality for this task that is slightly different than $\mathcal{F}_{\mathsf{BD}}$ specified above (Section 7.1). In particular, the ideal functionality $\mathcal{F}_{\mathsf{nrBD}}$ (Section 8.3.4) is not reusable (there is only one Turing machine queried by the client), and we allow the server to see the *output* of the client's desired computation, while still hiding the input x.

Theorem 8.21. *Assuming sub-exponentially secure post-quantum indistinguishability obfuscation and one-way functions,*[19] *there exists a two-message protocol* Π *that securely realizes (Definition 7.2) the functionality* $\mathcal{F}_{\mathsf{nrBD}}^{\mathsf{Del}}$ *against adversaries that corrupt the server, and where the client runs in time* $\mathrm{poly}(\lambda, |M|, |x|, \log t)$*.*

[19] As remarked in Section 4.3, it is known how to construct si$\mathcal{O}$ from sub-exponentially secure i$\mathcal{O}$.

Ideal Functionality $\mathcal{F}_{\mathsf{nrBD}}$

Parties: client C and server S, each with input the security parameter 1^λ.

- $\mathcal{F}_{\mathsf{nrBD}}$ receives a Turing machine M, an input x, and a step-size t from C.
- $\mathcal{F}_{\mathsf{nrBD}}$ computes $y = M^t(x)$ and sends (M, t, y) to S and y to C.

Figure 3: The ideal functionality for single-use blind delegation with certified deletion

Proof. The construction proceeds as follows. Let f be a one-way function. Suppose the client wants to delegate a Turing machine computation specified by M, x, t with one bit of output. They do the following.

- Sample $r_0, r_1 \leftarrow \{0,1\}^\lambda$ and define $s_0 := f(r_0)$ and $s_1 := f(r_1)$.
- Define Turing machine M' to take (x, r_0, r_1) as input, run $M(x)$ for t steps to obtain a bit b, and then output r_b.
- Sample $|\widetilde{M'}\rangle, \mathsf{vk} \leftarrow \mathsf{SRE\text{-}CD}(1^\lambda, M', (x, r_0, r_1), t)$ and output

$$|\widetilde{M'}\rangle, \mathsf{vk}, s_0, s_1.$$

The server then runs $r_b \leftarrow \mathsf{SRE\text{-}CD.Eval}(|\widetilde{M'}\rangle)$ coherently (that is, they only measure the output r_b), and then, if desired, runs $\mathsf{cert} \leftarrow \mathsf{SRE\text{-}CD.Del}(|\widetilde{M'}\rangle)$. The server returns (r_b, cert) to the client. If there exists b such that $s_b = f(r_b)$, then the client outputs b.

To argue security, we define the simulator to take as input $(M, t, b := M^t(x))$, sample r_0, r_1, compute $s_0 := f(r_0), s_1 := f(r_1)$, and run $|\widetilde{M'}\rangle, \mathsf{vk} \leftarrow \mathcal{S}(M', t, r_b)$, where $\mathcal{S}$ is the simulator for the SRE-CD scheme. Then, it runs the adversarial server on input $|\widetilde{M'}\rangle, \mathsf{vk}$. If it receives r such that $f(r) = s_b$, it instructs the ideal functionality to deliver the output, and otherwise it does not. If it receives cert such that $\mathsf{SRE\text{-}CD}(\mathsf{vk}, \mathsf{cert}) = \top$, it instructs the ideal functionality to deliver Deletion_Confirmed. It is straightforward to see that, due to the security of the one-way function f and the SRE-CD scheme, this satisfies Definition 7.2. In particular, note that the client does not need to keep any long-term secrets, so the tape **sec** is empty, and does not have to be simulated. □

Remark 8.22. *Even though $\mathcal{F}_{\mathsf{nrBD}}$ leaks the output y to the server, the client could always hide its "real" output from the server as follows. Suppose the client wants to delegate the computation of (M, t) on input $x \in \{0,1\}^n$. Consider the Turing machine (M', t) that takes $n+1$ bits (x, b) as input, and outputs $b \oplus M^t(x)$. Then the client can sample a random $b \leftarrow \{0,1\}$ and delegate M' on input (x, b). The server will only learn $b \oplus M^t(x)$, which is a uniformly random bit from their view.*

However, we remark that in this scenario, the client does *maintain a long-term secret, that is, the bit b. And we cannot explicitly leak b to the server. Thus, we can securely realize a variant of $\mathcal{F}_{\mathsf{nrBD}}$ where the ideal functionality* does not *deliver y to the server, but only under a variant of Definition 7.2 where the long-term secret tape* **sec** *is not leaked to the adversary.*

8.3.5 Oracles with certified deletion

In this section, we describe an application of our techniques in the "oracle model", where parties may have (quantum-accessible) oracle access to any classical functionality. We show how to implement an "oracle with certified deletion" for any classical functionality f (with polynomial description size). This primitive allows us to prepare an oracle $\widehat{f}$ that anyone may use to evaluate f. However, any adversary that queries $\widehat{f}$ some $\mathrm{poly}(\lambda)$ number of times and then produces a (classical and publicly-verifiable) certificate of deletion can no longer learn any information about f via $\widehat{f}$. That is, even given *unbounded* queries to $\widehat{f}$, the adversary will *never again* be able to learn anything about f, despite the fact that it was previously able to use $\widehat{f}$ to learn $f(x)$ for any inputs x of its choice.

Definition 8.23 (Oracle with certified deletion)**.** *An* oracle with certified deletion *consists of the following polynomial-time algorithms.*

- $\mathsf{Enc}(1^\lambda, f) \to (|\mathsf{ek}\rangle, \mathsf{vk}, \widehat{f})$*: The encode algorithm takes the security parameter* 1^λ *and a classical functionality* f *as input, and outputs a quantum evaluation key* $|\mathsf{ek}\rangle$*, a verification key* vk*, and a classical functionality* $\widehat{f}$*.*
- $\mathsf{Eval}^{\widehat{f}}(|\mathsf{ek}\rangle, x) \to y$*: The evaluation algorithm has oracle access to* $\widehat{f}$*, takes an evaluation key* $|\mathsf{ek}\rangle$ *and an* x *as input, and outputs* y*.*
- $\mathsf{Del}(|\mathsf{ek}\rangle) \to \mathsf{cert}$*: The deletion algorithm takes as input an evaluation key* $|\mathsf{ek}\rangle$ *and outputs a deletion certification* cert*.*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) \to \{\top, \bot\}$*: The verification algorithm takes as input the verification key* vk *and a deletion certificate* cert *and outputs* $\top$ *or* $\bot$*.*

It should satisfy the following properties.

- ***Correctness.*** *For any* λ*,* f*, and* x*,*

$$\Pr\left[\mathsf{Eval}^{\widehat{f}}(|\mathsf{ek}\rangle, x) = f(x) : |\mathsf{ek}\rangle, \mathsf{vk}, \widehat{f} \leftarrow \mathsf{Enc}(1^\lambda, f)\right] = 1.$$

- ***Correctness of deletion.*** *For any* λ *and* f*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top : \begin{array}{r} |\mathsf{ek}\rangle, \mathsf{vk}, \widehat{f} \leftarrow \mathsf{Enc}(1^\lambda, f) \\ \mathsf{cert} \leftarrow \mathsf{Del}(|\mathsf{ek}\rangle) \end{array}\right] = 1 - \mathrm{negl}(\lambda).$$

- ***Certified deletion.*** *There exists a simulator* $\mathcal{S}$ *such that for any adversary* $\mathcal{A}$ *with unbounded time and unbounded oracle queries, and any* f*,*

$$\left|\Pr[\mathsf{REAL}^{\mathcal{A}}(\lambda, f) = 1] - \Pr[\mathsf{IDEAL}^{\mathcal{A},\mathcal{S}}(\lambda, f) = 1]\right| = \mathrm{negl}(\lambda),$$

where the experiments are defined as follows. Given a security parameter λ *and a functionality* f*, the experiment* $\mathsf{REAL}^{\mathcal{A}}(\lambda, f)$ *proceeds as follows.*

  - *Sample* $(|\mathsf{ek}\rangle, \mathsf{vk}, \widehat{f}) \leftarrow \mathsf{Gen}(1^\lambda, f)$*.*

– *Run* $\mathcal{A}^{\widehat{f},\mathsf{Verify}(\mathsf{vk},\cdot)}(|\mathsf{ek}\rangle)$. *If* $\mathcal{A}$ *does not output a classical* cert *such that* $\mathsf{Verify}(\mathsf{vk},\mathsf{cert}) = \top$ *within its first* $\mathrm{poly}(\lambda)$ *oracle queries, then abort and output* $\bot$.

– *Otherwise, continue running* $\mathcal{A}^{\widehat{f},\mathsf{Verify}(\mathsf{vk},\cdot)}$ *until it halts and outputs a bit.*

Given a security parameter λ *and a functionality* f*, the experiment* $\mathsf{IDEAL}^{\mathcal{A},\mathcal{S}}(\lambda, f)$ *proceeds as follows.*

– *Sample* $(|\mathsf{ek}\rangle, \mathsf{vk}) \leftarrow \mathcal{S}(1^\lambda, |f|)$.

– *Run* $\mathcal{A}^{\mathcal{S}^f,\mathsf{Verify}(\mathsf{vk},\cdot)}(|\mathsf{ek}\rangle)$*, where each query made by* $\mathcal{A}$ *to* $\widehat{f}$ *is answered by* $\mathcal{S}^f$ *making a single query to* f*. If* $\mathcal{A}$ *does not output a classical* cert *such that* $\mathsf{Verify}(\mathsf{vk},\mathsf{cert}) = \top$ *within its first* $\mathrm{poly}(\lambda)$ *oracle queries, then abort and output* $\bot$.

– *Otherwise, once* $\mathcal{A}$ *outputs* cert *such that* $\mathsf{Verify}(\mathsf{vk},\mathsf{cert}) = \top$*, continue running* $\mathcal{A}^{\mathcal{S},\mathsf{Verify}(\mathsf{vk},\cdot)}$ *until it halts and outputs a bit. Note that in this final stage* $\mathcal{S}$ *does not have access to* f.

Now, we show how to construct an oracle with certified deletion. The querier will hold a quantum ciphertext containing the program to be evaluated. The oracle will run a program which checks that the ciphertext is intact, then decrypts it and evaluates it on the queried input.

Oracle Program

Hard-Coded Values. $\{S_i, T_i, \mathsf{u}_i\}_{i\in[\ell]}$, a bitstring $\widetilde{f} \in \{0,1\}^\ell$.

Input. Vectors $\{\mathsf{t}_i\}_{i\in[\ell]}$ and an input x.

1: **if** $\mathsf{t}_i \in T_i + \mathsf{u}_i$ **then**
2: $\quad \mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to and define $f'_i := \widetilde{f}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
3: $\quad$ Output $f'(x)$.

Oracle with Certified Deletion

$\mathsf{Enc}(1^\lambda, f)$**:**

1: Let $n = 4\lambda$, and for each $i \in [\ell]$, sample $S_i < T_i < \mathbb{F}_2^n$ such that $\dim(S_i) = n/2$ and $\dim(T_i) = 3n/4$, sample $\mathsf{v}_i, \mathsf{w}_i \leftarrow \mathsf{co}(S_i) \times \mathsf{co}(S_i^\perp)$, and let u_i be the coset of T_i that v_i belongs to.
2: For each $i \in [\ell]$, let $\widetilde{f}_i := f_i \oplus \langle \mathsf{v}_i, \mathbf{1}\rangle$, and define $\widetilde{f} := (\widetilde{f}_1, \ldots, \widetilde{f}_\ell)$.
3: Let $\hat{f}$ be the oracle program with hard-coded values $\{S_i, T_i, \mathsf{u}_i\}_{i\in[\ell]}$ and $\widetilde{f}$.
4: Output

$$|\mathsf{ek}\rangle := \{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[\ell]}, \quad \mathsf{vk} := \{S_i^\perp, \mathsf{w}_i\}_{i\in[\ell]}, \quad \widehat{f}.$$

$\mathsf{Eval}^{\widehat{f}}(|\mathsf{ek}\rangle, x)$**:** Measure $\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[\ell]}$ in the computational basis to obtain vectors $\{\mathsf{t}_i\}_{i\in[\ell]}$, and query $(\{\mathsf{t}_i\}_{i\in[\ell]}, x)$ to $\widehat{f}$ to obtain output y.
$\mathsf{Del}(|\mathsf{ek}\rangle)$**:** Measure $\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[\ell]}$ in the Hadamard basis to obtain vectors $\{\mathsf{z}_i\}_{i\in[\ell]}$, and output cert $:= \{\mathsf{z}_i\}_{i\in[\ell]}$.
$\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$**:** If $\mathsf{z}_i \in S_i^\perp + \mathsf{w}_i$ for all $i \in [\ell]$ then output $\top$ and otherwise output $\bot$.

Theorem 8.24. *The above construction is an oracle with certified deletion (Definition 8.23).*

Proof. Correctness and correctness of deletion follow immediately from the the description of the scheme, so it suffice to prove certified deletion.

We describe the simulator $\mathcal{S}$. Given 1^λ and a description length ℓ, $\mathcal{S}$ first samples $\{S_i, T_i, \mathsf{v}_i, \mathsf{w}_i, \mathsf{u}_i\}_{i\in[\ell]}$ as in the description of Enc, and sets $|\mathsf{ek}\rangle := \{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[\ell]}, \mathsf{vk} := \{S_i^\perp, \mathsf{w}_i\}_{i\in[\ell]}$. It answers $\widehat{f}$ queries as follows, where the classical functionality described below will be run in superposition over $\mathcal{A}$'s quantum query.

- For queries $(\{\mathsf{t}_i\}_{i\in[\ell]}, x)$ made before $\mathcal{A}$ produces cert, $\mathcal{S}$ checks that $\mathsf{t}_i \in T_i + \mathsf{u}_i$ for all $i \in [\ell]$, and outputs $\perp$ if not. Otherwise it queries f on x to obtain y and outputs y.
- For queries $(\{\mathsf{t}_i\}_{i\in[\ell]}, x)$ made after $\mathcal{A}$ produces cert, $\mathcal{S}$ answers using the program $P[\{S_i, T_i, \mathsf{u}_i\}_{i\in[\ell]}, \widetilde{f}]$, where $\widetilde{f} := (\langle \mathsf{v}_1, \mathbf{1}\rangle, \ldots, \langle \mathsf{v}_\ell, \mathbf{1}\rangle)$. Note that P is now completely independent of f.

We show via a sequence of hybrids that for any f, $\left|\Pr[\mathsf{REAL}^{\mathcal{A}}(\lambda, f) = 1] - \Pr[\mathsf{IDEAL}^{\mathcal{A},\mathcal{S}}(\lambda, f) = 1]\right| = \mathrm{negl}(\lambda)$.

- $\mathcal{H}_0$: This is $\mathsf{REAL}^{\mathcal{A}}(\lambda, f)$.
- $\mathcal{H}_1$: For each $i \in [\ell]$, sample R_i as a random superspace of $S_i^\perp$ of dimension $3n/4$, and let $\mathsf{x}_i \in \mathsf{co}(R_i)$ be such that $S_i^\perp + \mathsf{w}_i \subset R_i + \mathsf{x}_i$. Answer each of $\mathcal{A}$'s queries to Verify before it produces cert using $\mathsf{vk}' := \{R_i, \mathsf{x}_i\}_{i\in[\ell]}$.
- $\mathcal{H}_2$: Answer each of $\mathcal{A}$'s queries to $\widehat{f}$ before it produces cert using the strategy described in the simulator. That is, the $\{S_i\}_{i\in[\ell]}$ are not used to answer these queries, only $\{T_i, \mathsf{u}_i\}_{i\in[\ell]}$.
- $\mathcal{H}_3 - \mathcal{H}_{\ell+2}$: In $\mathcal{H}_i$ for $i \in [3, \ldots, \ell+2]$, switch the $(i-2)$'th bit of $\widetilde{f}$ from $f_{i-2} \oplus \langle \mathsf{v}_{i-2}, \mathbf{1}\rangle$ to $\langle \mathsf{v}_{i-2}, \mathbf{1}\rangle$.
- $\mathcal{H}_{\ell+3}$: Reverse the switch made in $\mathcal{H}_1$. That is, answer each of $\mathcal{A}$'s queries to Verify before it produces cert using $\mathsf{vk} := \{S_i^\perp, \mathsf{w}_i\}_{i\in[\ell]}$. This is the simulator described above.

Now we argue statistical indistinguishability between each pair of hybrids, which completes the proof.

- The difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is $\mathrm{negl}(\lambda)$, which follows from ℓ invocations of Theorem 4.11.
- The difference between $\mathcal{H}_1$ and $\mathcal{H}_2$ is $\mathrm{negl}(\lambda)$, which follows from ℓ invocations of Theorem 4.11.
- For each $i \in [3, \ldots, \ell+2]$, we can directly reduce the indistinguishability of $\mathcal{H}_{i-1}$ and $\mathcal{H}_i$ to Theorem 6.5 by noting that the only information obtained by $\mathcal{A}$ about $S_{i-2}, \mathsf{v}_{i-2}, \mathsf{w}_{i-2}$ via oracle queries to $\widetilde{f}$ and Verify are the random superspaces $T_{i-2}, \mathsf{u}_{i-2}, R_{i-2}, \mathsf{x}_{i-2}$.
- The difference between $\mathcal{H}_{\ell+2}$ and $\mathcal{H}_{\ell+3}$ is $\mathrm{negl}(\lambda)$, which follows from ℓ invocations of Theorem 4.11.

□

9 Functional Encryption with Certified Deletion

We consider two notions of functional encryption with certified deletion. One allows the secret key to be deleted, while the other allows ciphertexts to be deleted. In functional encryption with *secret key certified deletion*, a secret key sk_f can be used to learn $f(x)$ from any ciphertext $\mathsf{Enc}(x)$ until sk_f is deleted, then it can no longer be used to learn anything. This notion was previously studied by [KN22]. In functional encryption with *ciphertext certified deletion*, if a key sk_f is received before $c = \mathsf{Enc}(x)$ is deleted, then it can be used to learn $f(x)$. However, no more information about x can be learned after c is deleted, even if new secret keys are received.

9.1 Definitions

Functional Encryption. A functional encryption scheme $\mathsf{FE} = (\mathsf{FE\text{-}Setup}, \mathsf{FE\text{-}KeyGen}, \mathsf{FE\text{-}Enc}, \mathsf{FE\text{-}Dec})$ for a family of message spaces $\{\mathcal{X}_n\}$, a family of output spaces $\{\mathcal{Y}_n\}$ and a family of functions $\mathcal{F}$ consists of the following polynomial time algorithms:

- $\mathsf{FE\text{-}Setup}(1^\lambda)$. The setup algorithm takes as input the security parameter λ and outputs a master public key-secret key pair $(\mathsf{pp}, \mathsf{msk})$.
- $\mathsf{FE\text{-}Enc}(\mathsf{pp}, x) \to \mathsf{ct}$. The encryption algorithm takes as input a message $x \in \mathcal{X}_n$ and the master public key pp. It outputs a ciphertext ct.
- $\mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input a function $f \in \mathcal{F}_\lambda$ and the master secret key msk. It outputs a function secret key sk_f.
- $\mathsf{FE\text{-}Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$. The decryption algorithm takes as input a secret key sk_f and a ciphertext ct. It outputs a string $y \in \mathcal{Y}_\lambda$ or $\perp$.

Definition 9.1. *(Correctness) A functional encryption scheme* FE *for* $\mathcal{F}$ *is correct if for all* $f \in \mathcal{F}_\lambda$ *and all* $x \in \mathcal{X}_\lambda$

$$\Pr\left[\mathsf{FE\text{-}Dec}(\mathsf{sk}_f, \mathsf{FE\text{-}Enc}(\mathsf{pp}, x)) = f(x) : \begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{FE\text{-}Setup}(1^\lambda) \\ \mathsf{sk}_f \leftarrow \mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f)\end{array}\right] = 1$$

where the probability is over the random coins of $\mathsf{FE\text{-}Setup}, \mathsf{FE\text{-}Enc}, \mathsf{FE\text{-}KeyGen}$ *and* $\mathsf{FE\text{-}Dec}$.

Security. We define (standard) security of functional encryption using the following game (Adaptive-IND) between a challenger and an adversary.

- **Setup Phase:** The challenger generates $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{FE\text{-}Setup}(1^\lambda)$ and then hands over the master public key pp to the adversary.
- **Key Query Phase 1:** The adversary makes function secret key queries by submitting functions $f \in \mathcal{F}_\lambda$. The challenger responds by giving the adversary the corresponding function secret key $\mathsf{sk}_f \leftarrow \mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f)$.
- **Challenge Phase:** The adversary chooses two messages M_0, M_1 of the same size (each in $\mathcal{X}_\lambda$) such that for all queried functions f in the key query phase, it holds that $f(M_0) = f(M_1)$. The challenger selects a random bit $\mathsf{b} \in \{0, 1\}$ and sends a ciphertext $\mathsf{ct} \leftarrow \mathsf{FE\text{-}Enc}(\mathsf{pp}, M_b)$ to the adversary.

- **Key Query Phase 2:** The adversary may submit additional key queries $f \in \mathcal{F}_\lambda$ as long as they do not violate the constraint described above.
- **Guess:** The adversary submits a guess b' and wins if $b' = b$. The adversary's advantage in this game is defined to be $2 \cdot |\Pr[b = b'] - 1/2|$.

Definition 9.2 (Adaptive-IND Security)**.** *A functional encryption scheme* FE *is adaptively secure if all PPT adversaries have at most a negligible advantage in the* Adaptive-IND *security game.*

Multi-input Functional Encryption. A (public-key) multi-input functional encryption scheme extends the definition of functional encryption to consider n-ary functions. For our construction, we will rely on a multi-input FE scheme that supports functions of arity two. The syntax of MIFE remains the same as above except that we have *two* encryption algorithms $\mathsf{MIFE\text{-}Enc}_1$ and $\mathsf{MIFE\text{-}Enc}_2$, one for each arity, and the decryption algorithm MIFE-Dec accepts *two* ciphertext as input, along with a functional secret key.

Definition 9.3. *(Correctness) A multi-input functional encryption scheme* MIFE *for $\mathcal{F}$ is correct if for all $f \in \mathcal{F}_\lambda$, all $x \in \mathcal{X}_\lambda$ and $y \in \mathcal{Y}_\lambda$,*

$$\Pr\left[\begin{array}{c}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{MIFE\text{-}Setup}(1^\lambda) \\ \mathsf{sk}_f \leftarrow \mathsf{MIFE\text{-}KeyGen}(\mathsf{msk}, f) \\ \mathsf{MIFE\text{-}Dec}(\mathsf{sk}_f, \mathsf{MIFE\text{-}Enc}_1(\mathsf{pp}, x), \mathsf{MIFE\text{-}Enc}_2(\mathsf{pp}, y)) = f(x, y)\end{array}\right] = 1$$

where the probability is over the random coins of MIFE-Setup, MIFE-Enc, MIFE-KeyGen *and* MIFE-Dec*.*

Security. We will require the following notion of security (Adaptive-IND-MIFE) from the MIFE scheme.

- **Setup Phase:** The challenger generates $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{FE\text{-}Setup}(1^\lambda)$ and then hands over the master public key pp to the adversary.
- **Key Query Phase 1:** The adversary makes function secret key queries by submitting functions $f \in \mathcal{F}_\lambda$. The challenger responds by giving the adversary the corresponding function secret key $\mathsf{sk}_f \leftarrow \mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f)$.
- **Challenge Phase:** The adversary chooses two messages M_0, M_1 of the same size (each in $\mathcal{X}_\lambda$) such that for all queried functions f in the key query phase, and every $y \in \mathcal{Y}$ it holds that $f(M_0, y) = f(M_1, y)$. The challenger selects a random bit $\mathsf{b} \in \{0, 1\}$ and sends a ciphertext $\mathsf{ct} \leftarrow \mathsf{FE\text{-}Enc}_1(\mathsf{pp}, M_b)$ to the adversary.
- **Key Query Phase 2:** The adversary may submit additional key queries $f \in \mathcal{F}_\lambda$ as long as they do not violate the constraint described above.
- **Guess:** The adversary submits a guess b' and wins if $b' = b$. The adversary's advantage in this game is defined to be $2 \cdot |\Pr[b = b'] - 1/2|$.

Definition 9.4 (Adaptive-IND-MIFE Security)**.** *A multi-input functional encryption scheme* MIFE *is adaptively secure if all PPT adversaries have at most a negligible advantage in the* Adaptive-IND-MIFE *security game.*

Remark 9.5. *We note that* MIFE *satisfying the above (adaptive) definition can be realized from sub-exponentially secure one-way functions and sub-exponentially secure* i$\mathcal{O}$ *following [GGG+14, GJO16, BPW16], and sub-exponentially secure* i$\mathcal{O}$ *can itself be based on sub-exponentially secure FE [AJ15, BV15]. In a weaker selective setting, where the challenge phase occurs before all other phases, this definition can be realized from polynomially secure* i$\mathcal{O}$ *[GGG+14]. We also observe that* MIFE *satisfying the above definition implies* i$\mathcal{O}$*, where to obfuscate a program, we simply encrypt the description of the program via* $\mathsf{MIFE\text{-}Enc}_1$*, and output the functional key for an (appropriately large) universal circuit. This obfuscated circuit can be evaluated on any input x by encrypting it using* $\mathsf{MIFE\text{-}Enc}_2$*, and running functional decryption.*

Notions of Certified Deletion

Certified deletion for ciphertexts. A (quantum) functional encryption scheme with certified deletion for ciphertexts modifies the above syntax so that $\mathsf{FE\text{-}Enc}(\mathsf{pp}, x)$ outputs a quantum ciphertext $|\mathsf{ct}\rangle$ and (classical) verification key vk, and includes the following algorithms.

- $\mathsf{Del}(|\mathsf{ct}\rangle) \to \mathsf{cert}$. The deletion algorithm takes as input a quantum ciphertext ct and outputs a (classical) deletion certificate cert.
- $\mathsf{Verify}(|\mathsf{ct}\rangle, \mathsf{vk}) \to \{\top, \bot\}$ takes as input a (classical) ciphertext and verification key, and outputs either $\top$ or $\bot$.

It additionally satisfies the following correctness of deletion property.

Definition 9.6. *(Correctness of Deletion) A functional encryption scheme* FE *for $\mathcal{F}$ satisfies correctness of deletion if for all $x \in \mathcal{X}_\lambda$*

$$\Pr\left[\begin{array}{c}(\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{FE\text{-}Setup}(1^\lambda)\\ (|\mathsf{ct}\rangle, \mathsf{vk}) \leftarrow \mathsf{FE\text{-}Enc}(\mathsf{pp}, x)\\ \mathsf{cert} \leftarrow \mathsf{Del}(|\mathsf{ct}\rangle)\\ \mathsf{Verify}(\mathsf{cert},\mathsf{vk}) = \top\end{array}\right] = 1$$

where the probability is over the random coins of $\mathsf{FE\text{-}Setup}, \mathsf{FE\text{-}Enc}, \mathsf{Del}$ *and* Verify.

Finally, it also satisfies a certified deletion property.

Definition 9.7. *(FE certified everlasting adaptive security for ciphertexts.) For all QPT adversaries $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$\mathsf{TD}\left(\mathsf{EV\text{-}EXP}_0(\mathcal{A}_\lambda), \mathsf{EV\text{-}EXP}_1(\mathcal{A}_\lambda)\right) = \mathrm{negl}(\lambda),$$

where, given an adversary $\mathcal{A}$, the experiment $\mathsf{EV\text{-}EXP}_b(\mathcal{A})$ is defined as follows.
$\mathsf{EV\text{-}EXP}_b(\mathcal{A}_\lambda)$*:*

- **Setup Phase:** *The challenger generates $(\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{FE\text{-}Setup}(1^\lambda)$ and then hands over the master public key* pp *to the adversary.*
- **Key Query Phase 1:** *$\mathcal{A}$ makes function secret key queries by submitting functions $f \in \mathcal{F}_\lambda$. The challenger responds by giving the adversary the corresponding function secret key $\mathsf{sk}_f \leftarrow \mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f)$.*
- **Challenge Phase:** *$\mathcal{A}$ chooses two messages M_0, M_1 of the same size (each in $\mathcal{X}_\lambda$) such that for all queried functions f in the key query phase, it holds that $f(M_0) = f(M_1)$. The challenger selects a random bit $\mathsf{b} \in \{0,1\}$ and sends a ciphertext $\mathsf{ct} \leftarrow \mathsf{FE\text{-}Enc}(\mathsf{pp}, M_b)$ to the adversary.*

- **Key Query Phase 2:** *$\mathcal{A}$ may submit additional key queries $f \in \mathcal{F}_\lambda$ as long as they do not violate the constraint described above.*
- **Deletion Phase:** *At some point, $\mathcal{A}$ sends a certificate of deletion* cert.

  Let ρ denote the left-over state of $\mathcal{A}$. If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \top$*, output ρ and otherwise output $\bot$.*

We say it is publicly verifiable if this holds even when $\mathcal{A}$ also receives the vk *which was generated alongside* ct.

It is also possible to define a certified everlasting selective variant of the definition above, where the challenge phase occurs before all other phases in the experiment.

Key revocation. A (quantum) functional encryption scheme with key revocation modifies the syntax of a functional encryption scheme similarly to that of a functional encryption scheme with certified deletion ciphertexts. However, instead of Del taking a ciphertext as input, it takes a secret key $|\mathsf{sk}_f\rangle$ as input. It must also satisfy deletion correctness (defined similarly to ciphertext deletion correctness) and secret key certified deletion.

Definition 9.8 (FE computational certified deletion adaptive security for secret keys.)**.** *A functional encryption scheme, augmented as above, has key revocation if the advantage of every QPT adversary in the following game is* $\mathrm{negl}(\lambda)$*:*

- ***Setup Phase:*** *The challenger samples* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{FE\text{-}Setup}(1^n)$ *and sends* pk *to the adversary.*
- ***Query Phase 1:*** *The following query phase is repeated a polynomial number of times:*
  1. *The adversary adaptively submits a query $f_i \in \mathcal{F}(n)$.*
  2. *The challenger samples* $(\mathsf{sk}_{f_i}, \mathsf{vk}_i) \leftarrow \mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f_i)$ *and sends* sk_{f_i} *to the adversary.*
- ***Deletion Phase:*** *Let n be the number of iterations in the query phase. The adversary sends a list of deletion proofs* $\mathsf{cert}_1, \ldots, \mathsf{cert}_n$*, along with two messages m_0 and m_1. For any secret key* sk_{f_i} *they do not wish to delete, they may send* $\mathsf{cert}_i = \bot$*.*
- ***Challenge Phase:*** *The challenger checks the deletion proofs. If $f_i(m_0) = f_i(m_1)$ for every f_i such that* $\mathsf{Verify}(\mathsf{cert}_i, \mathsf{vk}_i) = \bot$*, then sample a random bit b and send* $\mathsf{Enc}(\mathsf{pk}, m_b)$ *to the adversary.*
- ***Query Phase 2:*** *The following query phase is repeated a polynomial number of times:*
  1. *The adversary adaptively submits a query $f_i \in \mathcal{F}(n)$.*
  2. *If $f_i(m_0) = f_i(m_1)$, the challenger samples* $(\mathsf{sk}_{f_i}, \mathsf{vk}_i) \leftarrow \mathsf{FE\text{-}KeyGen}(\mathsf{msk}, f_i)$ *and sends* sk_{f_i} *to the adversary.*
- *The adversary outputs a bit b' and wins if $b' = b$.*

We say the functional encryption scheme has selective *secret key certified deletion if this holds in the game where adversary must declare the challenge messages m_0 and m_1* before *the challenger samples* $(\mathsf{pk}, \mathsf{sk})$.

We say it has public verifiability if this holds even when the adversary also receives vk_i *during the query phases.*

We note that the above definition only guarantees computational security after deletion. However, this is unavoidable. Any functional encryption scheme is also a public key encryption scheme, so the ciphertext itself can only be computationally hiding, even *without* access to any secret keys.

9.2 Some Preliminaries

In Appendix A, we prove a variant of subspace-hiding security for functional encryption, building on subspace-hiding obfuscation [Zha19]. Our proof makes use of the following corollary of the lemma we prove in Appendix A.

Corollary 9.9. *(Subspace-Hiding for Multi-input Functional Encryption) Any multi-input functional encryption scheme* $(\mathsf{MIFE\text{-}Setup}, \mathsf{MIFE\text{-}KeyGen}, \mathsf{MIFE\text{-}Enc}_1, \mathsf{MIFE\text{-}Enc}_2, \mathsf{MIFE\text{-}Dec})$ *of arity two (Definition 9.4) satisfies the following.*

$$\Pr\left[b'=b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ (\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{MIFE\text{-}Setup}(1^\lambda) \\ M \leftarrow \mathcal{A}^{\mathsf{MIFE\text{-}KeyGen}(\mathsf{msk},g(\cdot))}(\mathsf{pp}) \\ \mathsf{ct} \leftarrow \mathsf{MIFE\text{-}Enc}_2(S,T,\mathsf{u},\mathsf{v},M,0,0) \text{ if } b=0 \\ \mathsf{ct} \leftarrow \mathsf{MIFE\text{-}Enc}_2(S,0,0,\mathsf{v},M,1,0) \text{ if } b=1 \\ b' \leftarrow \mathcal{A}^{\mathsf{MIFE\text{-}KeyGen}(\mathsf{msk},g(\cdot))}(\mathsf{pp},\mathsf{sk}_f,\mathsf{ct},S,\mathsf{v}) \end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is over the randomness of sampling $S < T < \mathbb{F}_2^n$ *such that* $\dim(S) \leq \dim(T)$ *and both dimensions are in* $[n/2, 3n/4]$ *for* $n = 4\lambda$, $\mathsf{v},\mathsf{w} \leftarrow \mathsf{co}(S) \times \mathsf{co}(S^\perp)$, *and letting* u *be the coset of* T *that* v *belongs to, and* $g(f)$ *is an appropriately defined function that obtains input* $(\mathsf{t},(S,T,\mathsf{u},\mathsf{v},M,c,\mathsf{td}))$ *and parses* $\mathsf{td} = (c_1,i,y)$. *Then, if* $c_1 = 0$, *it does the following:*

- *If* $c = 0$, *if* $\mathsf{t} \notin T + \mathsf{u}$, *then abort and output* $\perp$, *and otherwise output* $f(M)$.
- *If* $c = 1$, *if* $\mathsf{t} \notin S + \mathsf{v}$, *then abort and output* $\perp$, *and otherwise output* $f(M)$.

Remark 9.10. *We note that in the corollary above,* g_f *is also well defined on values of* $c_1 \neq 0$, *but we skip these details in the statement of the Corollary in this section, and defer them to the expanded version presented in Appendix A.*

9.3 Construction of Functional Encryption with Certified Deletion for Ciphertexts

Let $\mathsf{MIFE} = (\mathsf{MIFE\text{-}Setup}, \mathsf{MIFE\text{-}KeyGen}, \mathsf{MIFE\text{-}Enc}_1, \mathsf{MIFE\text{-}Enc}_2, \mathsf{MIFE\text{-}Dec})$ denote a public-key multi-input functional encryption scheme of arity two, satisfying Definition 9.4.

We build $\mathsf{FE.CD} = (\mathsf{FE.CD\text{-}Setup}, \mathsf{FE.CD\text{-}KeyGen}, \mathsf{FE.CD\text{-}Enc}, \mathsf{FE.CD\text{-}Dec})$ as follows.

FE with Certified Deletion for Ciphertexts

$\mathsf{FE.CD\text{-}Setup}(1^\lambda)$: Output $(\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{MIFE\text{-}Setup}(1^\lambda)$.

$\mathsf{FE.CD\text{-}Enc}(\mathsf{pp}, m)$:

1: Let $n = 4\lambda$, let $\ell = |m|$, and for $i \in [2\ell+1]$, sample $S_i < T_i < \mathbb{F}_2^n$ such that $\dim(S_i) = n/2$ and $\dim(T_i) = 3n/4$, sample $\mathsf{v}_i, \mathsf{w}_i \leftarrow \mathsf{co}(S_i) \times \mathsf{co}(S_i^\perp)$, and let u_i be the coset of T_i that v_i belongs to.
2: Let $\widehat{m} := (0, m, 0^\ell)$, and for all $i \in [2\ell+1]$, define $\widetilde{m}_i := \widehat{m}_i \oplus \langle \mathsf{v}_i, \mathbf{1} \rangle$. Define $\widetilde{m} := (\widetilde{m}_1, \ldots, \widetilde{m}_{2\ell+1})$.

3: Let $|\mathsf{ct}_1\rangle = \{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]}$ and $\mathsf{ct}_2 = \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)$.
4: Output $|\mathsf{ct}\rangle = |\mathsf{ct}_1\rangle, \mathsf{ct}_2$ and the certified deletion verification key $\mathsf{vk} := \left\{i\mathcal{O}\left(P_{S_i^\perp+\mathsf{w}_i}\right)\right\}_{i\in[2\ell+1]}$.

FE.CD-KeyGen(msk, f): Output $\mathsf{sk}_f = \mathsf{MIFE\text{-}KeyGen}(\mathsf{msk}, g_f)$, where g_f is the following function.

1: Take vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$ as the first input, and parse the second input as the set $\left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2\ell+1]}, c\right)$ and a final message M.
2: If $M \neq \perp$, output $f(M)$ and end. Otherwise, continue[a].
3: **for** each $i \in [2\ell+1]$ such that $c_i = 0$ **do**
4: If $\mathsf{t}_i \notin T_i + \mathsf{u}_i$, then abort and output $\perp$, and otherwise let $\mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to, and define $m'_i := \widetilde{m}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
5: **for** each $i \in [2\ell+1]$ such that $c_i = 1$ **do**
6: If $\mathsf{t}_i \notin S_i + \mathsf{v}_i$, then abort and output $\perp$, and otherwise let $\mathsf{v}'_i \in \mathsf{co}(S_i)$ be the coset of S_i that t_i belongs to, and define $m'_i := \widetilde{m}_i \oplus \langle \mathsf{v}'_i, \mathbf{1}\rangle$.
7: Parse $m'_1, \ldots, m'_{2\ell+1}$ as (b, m_0, m_1). Output $f(m_b)$.

FE.CD-Dec$(\mathsf{pp}, \mathsf{sk}_f, |\mathsf{ct}\rangle)$

1: Parse $|\mathsf{ct}\rangle = |\mathsf{ct}_1\rangle, \mathsf{ct}_2$.
2: Measure $|\mathsf{ct}_1\rangle = \{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]}$ in the computational basis to obtain vectors $\{\mathsf{t}_i\}_{i\in[2\ell+1]}$. Let $\mathsf{ct}' = \mathsf{FE\text{-}Enc}_1(\mathsf{pp}, \{\mathsf{t}_i\}_{i\in[2\ell+1]})$
3: Output the result of $\mathsf{FE\text{-}Dec}(\mathsf{ct}', \mathsf{ct}_2)$.

Del$(|\mathsf{ct}\rangle)$: Measure $\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\in[2\ell+1]}$ in the Hadamard basis to obtain vectors $\{\mathsf{z}_i\}_{i\in[2\ell+1]}$, and output $\mathsf{cert} := \{\mathsf{z}_i\}_{i\in[2\ell+1]}$.

Verify$(\mathsf{vk}, \mathsf{cert})$: If $i\mathcal{O}\left(P_{S_i^\perp+\mathsf{w}_i}\right)(\mathsf{z}_i) = 1$ for all $i \in [2\ell+1]$ then output $\top$ and otherwise output $\perp$.

[a] In the rest of this section, we will often omit explicitly writing the last part of the input, this should be taken to denote that the final message M is parsed as $\perp$ and this "if" statement is skipped. Furthermore, g_f contains additional "if" statements matching the ones in Corollary 9.9 that will be used when we rely on Corollary 9.9 in the proof, but we skip these here for notational convenience.

Theorem 9.11. *Assuming post-quantum public-key multi-input functional encryption adaptively (resp., selectively) secure against unbounded collusions (Definition 9.4), there exists public-key functional encryption with ciphertext certified everlasting adaptive (resp., selective) security (Definition 9.7) with publicly verifiable deletion.*

Proof. Correctness of decryption and correctness of deletion are immediate from the scheme. Thus, it remains to show computational security and certified everlasting security. We prove this in the adaptive setting below; a proof in the selective setting follows similarly.

Computational security. We have the following hybrids, where changes between subsequent hybrids are colored in red.

- $\mathcal{H}_0$: This corresponds to Exp_0 where $(|\mathsf{ct}\rangle, \mathsf{vk}) \leftarrow \mathsf{FE.CD\text{-}Enc}(\mathsf{pp}, M_0)$.
- $\mathcal{H}_1$: Generate ciphertext ct_2 as $\mathsf{MIFE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)$.
- $\mathcal{H}_2$: Generate ciphertext ct_2 as $\mathsf{MIFE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2\ell+1]}, 1^{2\ell+1}\right)\right)$.
- $\mathcal{H}_3$: Replace $\widehat{m} = (0, M_0, 0^\ell)$ with $\widehat{m} = (0, M_1, 0^\ell)$.

- $\mathcal{H}_4$: Generate ciphertext ct_2 as $\mathsf{MIFE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)$.
- $\mathcal{H}_5$: Generate ciphertext ct_2 as $\mathsf{MIFE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)$. This corresponds to Exp_1 where $\left(|\mathsf{ct}\rangle, \mathsf{vk}\right) \leftarrow \mathsf{FE.CD\text{-}Enc}(\mathsf{pp}, M_1)$.

Indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ and between $\mathcal{H}_4$ and $\mathcal{H}_5$ follows from MIFE security, because the v_i values are never used. Indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$, and $\mathcal{H}_3$ and $\mathcal{H}_4$ follows from Corollary 9.9. Finally, indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from the security of FE due to the fact that $f(M_0) = f(M_1)$.

Certified everlasting security. We will switch an encryption of M_0 to an encryption of M_1, by changing one bit of the string $\widehat{m}$ in the construction at a time, and argue that each switch is *statistically close* conditioned on the adversary producing a successful deletion certificate.

- $\mathcal{H}_0$: This is the certified everlasting security game with $\left|\widetilde{\mathsf{ct}}\right\rangle, \mathsf{vk} \leftarrow \mathsf{FE.CD\text{-}Enc}(1^\lambda, M_0)$.
- $\mathcal{H}_1 - \mathcal{H}_\ell$: In hybrid $\mathcal{H}_i$ for $i \in [1, \ldots, \ell]$, we switch the $\ell + 1 + i$'th bit of $\widehat{m}$ to the i'th bit of the description of M_1. That is, in $\mathcal{H}_\ell$, the string $\widehat{m} = (0, M_0, M_1)$.
- $\mathcal{H}_{\ell+1}$: Switch the first bit of $\widehat{m}$ to 1. So, now $\widehat{m} = (1, M_0, M_1)$.
- $\mathcal{H}_{\ell+2} - \mathcal{H}_{2\ell+1}$: In hybrid $\mathcal{H}_i$ for $i \in [\ell + 2, \ldots, 2\ell + 1]$, we switch the $i - \ell'$th bit of $\widehat{m}$ to the $i - \ell - 1$'th bit of M_1. That is, in $\mathcal{H}_{2\ell+1}$, the string $\widehat{m} = (1, M_1, M_1)$.
- $\mathcal{H}_{2\ell+2}$: Switch the first bit of $\widehat{m}$ to 0. So, now $\widehat{m} = (0, M_1, M_1)$.
- $\mathcal{H}_{2\ell+3} - \mathcal{H}_{3\ell+2}$: In hybrid $\mathcal{H}_i$ for $i \in [2\ell + 3, 3\ell + 2]$, we switch the $i - \ell - 1$'th bit of $\widehat{m}$ to 0. That is, in $\mathcal{H}_{3\ell+2}$, the string $\widehat{m} = (0, M_1, 0^\ell)$. Note that this is exactly the certified everlasting security game with $\left|\widetilde{ct}\right\rangle, \mathsf{vk} \leftarrow \mathsf{i}\mathcal{O}\text{-}\mathsf{CD}(1^\lambda, M_1)$.

The proof follows by combining the following claims.

Claim 9.12. *For all $i \in [1, \ldots, \ell]$, $\mathsf{TD}(\mathcal{H}_{i-1}, \mathcal{H}_i) = \mathrm{negl}(\lambda)$.*

Proof. We will reduce this claim to Theorem 6.5. To do so, we must define a distribution $\mathcal{Z}$ and argue that it is *subspace-hiding* according to Definition 6.4. Defining $i^* := \ell + 1 + i$, we note that the output of FE.CD-Enc in $\mathcal{H}_{i-1}/\mathcal{H}_i$ can be written as

$$\left|(S_{i^*})_{\mathsf{v}_{i^*}, \mathsf{w}_{i^*}}\right\rangle, \left(\{|(S_i)_{\mathsf{v}_i, \mathsf{w}_i}\rangle\}_{i\neq i^*}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)\right)$$

where in $\mathcal{H}_{i-1}$, $\widetilde{m}_{i^*} = 0 \oplus \langle \mathsf{v}_{i^*}, \mathbf{1}\rangle$ and in $\mathcal{H}_i$, $\widetilde{m}_{i^*} = (M_1)_i \oplus \langle \mathsf{v}_{i^*}, \mathbf{1}\rangle$. Thus, we must show that the following distribution $\mathcal{Z}_\lambda$ is subspace-hiding.

$\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$:

- Set $S_{i^*} := S, T_{i^*} := T, \mathsf{u}_{i^*} := \mathsf{u}, \mathsf{w}_{i^*} := \mathsf{w}_i$, and $\widetilde{m}_{i^*} := b'$.
- For $i \in [2\ell + 1] \setminus \{i^*\}$, sample $S_i < T_i < \mathbb{F}_2^n$ uniformly at random such that $\dim(S) = n/2$ and $\dim(T) = 3n/4$, sample $\mathsf{v}_i \leftarrow \mathsf{co}(S_i) \cap T_i$, and let u_i be the coset of T_i that v_i belongs to.

- Sample the bits $\widetilde{m}_i$ for $i \neq i^*$ as they are sampled in $\mathcal{H}_{i-1}$ and $\mathcal{H}_i$, and output

$$\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq i^*}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right), \left\{\mathsf{i}\mathcal{O}\left(P_{S_i^\perp+\mathsf{w}_i}\right)\right\}_{i\in[2\ell+1]}.$$

We will proceed via a sequence of hybrids.

- $\mathcal{H}_0(S, T, \mathsf{u}, \mathsf{w}, b')$: This is $\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$.
- $\mathcal{H}_1(S, T, \mathsf{u}, \mathsf{w}, b')$: Same as $\mathcal{H}_0$, except that the ciphertext is

$$\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq i^*}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2\ell+1]}, 1^{\ell+1}0^\ell\right)\right)$$

- $\mathcal{H}_2(S, T, \mathsf{u}, \mathsf{w}, b')$: Same as $\mathcal{H}_1$, except that the ciphertext is

$$\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq i^*}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[\ell+1]}, \{0, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[\ell+2,2\ell+1]}, 1^{\ell+1}0^\ell, M_0\right)\right)$$

- $\mathcal{H}_3(S, T, \mathsf{u}, \mathsf{w}, b')$: Same as $\mathcal{H}_2$, except that we sample R_{i^*} as a uniformly random superspace of $S_{i^*}^\perp$ of dimension $3n/4$, define $\mathsf{x}_{i^*} \in \mathsf{co}(R_{i^*})$ so that $S_{i^*}^\perp + \mathsf{w}_{i^*} \subset R_{i^*} + \mathsf{x}_{i^*}$, and set the verification key to be $\mathsf{i}\mathcal{O}\left(P_{R_{i^*}+\mathsf{x}_i}\right)$ in place of $\mathsf{i}\mathcal{O}\left(P_{S_{i^*}^\perp+\mathsf{w}_i}\right)$. Note that this distribution can be prepared just given $(R, T, \mathsf{u}, \mathsf{x}, b')$ as defined in Definition 6.4, and thus can be considered the simulated distribution.

Now, the indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_1$ follows by MIFE security and repeated application of subspace-hiding functional encryption (Corollary 9.9) for each $i \in [1, \ldots, \ell+1]$. Next, the indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the security of MIFE. Indeed, note that in $\mathcal{H}_1$, the functional keys will always either abort or unmask the first $\ell+1$ bits of $\widetilde{m}$ to $(0, M_0)$, which means that the final step of the program will always output $f(M_0)$. Finally, the indistinguishability of $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from Corollary 4.10. □

Claim 9.13. $\mathsf{TD}(\mathcal{H}_\ell, \mathcal{H}_{\ell+1}) = \mathrm{negl}(\lambda)$.

Proof. We will again reduce this claim to Theorem 6.5. Note that the output of $\mathsf{i}\mathcal{O}$-CD in $\mathcal{H}_\ell/\mathcal{H}_{\ell+1}$ can be written as

$$|(S_1)_{\mathsf{v}_1,\mathsf{w}_1}\rangle, \left(\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq 1}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)\right)$$

$\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$:

- Set $S_1 := S, T_1 := T, \mathsf{u}_1 := \mathsf{u}, \mathsf{w}_1 := \mathsf{w}$, and $\widetilde{m}_1 := b'$.
- For $i \in [2\ell+1] \setminus \{1\}$, sample $S_i < T_i < \mathbb{F}_2^n$ uniformly at random such that $\dim(S) = n/2$ and $\dim(T) = 3n/4$, $\mathsf{v}_i \leftarrow \mathsf{co}(S_i) \cap T_i$, and let u_i be the coset of T_i that v_i belongs to.
- Sample the bits $\widetilde{m}_i$ for $i \neq 1$ as they are sampled in $\mathcal{H}_{i-1}$ and $\mathcal{H}_i$, and output

$$|(S_1)_{\mathsf{v}_1,\mathsf{w}_1}\rangle, \left(\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq 1}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, 0, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0^{2\ell+1}\right)\right)\right)$$

We will proceed via a sequence of hybrids.

- $\mathcal{H}_0(S, T, \mathsf{u}, \mathsf{w}, b')$: This is $\mathcal{Z}_\lambda(S, T, \mathsf{u}, \mathsf{w}, b')$.
- $\mathcal{H}_1(S, T, \mathsf{u}, \mathsf{w}, b')$: Same as $\mathcal{H}_0$, except that we output

$$\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq 1}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(\{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2\ell+1]}, 0||1^{2\ell}\right)\right)$$

- $\mathcal{H}_2(S, T, \mathsf{u}, \mathsf{w}, b')$: Same as $\mathcal{H}_1$, except that we output

$$\{|(S_i)_{\mathsf{v}_i,\mathsf{w}_i}\rangle\}_{i\neq 1}, \mathsf{FE\text{-}Enc}_2\left(\mathsf{pp}, \left(0, T_1, \mathsf{u}_1, 0, \widetilde{m}_1, \{S_i, T_i, \mathsf{u}_i, \mathsf{v}_i, \widetilde{m}_i\}_{i\in[2,2\ell+1]}, 0||1^{2\ell}, M_0\right)\right)$$

- $\mathcal{H}_3(S, T, \mathsf{u}, \mathsf{w}, b')$: Same as $\mathcal{H}_2$, except that we sample R_1 as a uniformly random superspace of $S_1^\perp$ of dimension $3n/4$, define $\mathsf{x}_1 \in \mathsf{co}(R_1)$ so that $S_1^\perp + \mathsf{w}_1 \subset R_1 + \mathsf{x}_1$, and use $(P_{R_1+\mathsf{x}_1})$ in place of $\left(P_{S_1^\perp+\mathsf{w}_1}\right)$ to verify certificates. Note that this distribution can be prepared just give $(R, T, \mathsf{u}, \mathsf{x}, b')$ as defined in Definition 6.4, and thus can be considered the simulated distribution.

Now, the indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_1$ follows by security of MIFE and by repeated application of Corollary 9.9 for each $i \in [2, \ldots, 2\ell + 1]$. Next, the indistinguishability of $\mathcal{H}_1$ and $\mathcal{H}_2$ follows from the security of MIFE. Indeed, note that in $\mathcal{H}_1$, the program will always either abort or unmask the $\widetilde{m}$ as (b, M_0, M_1) for some arbitrary bit b. Since $f(M_0) = f(M_1)$, this means that the last step always outputs $f(M_0)$. Finally, the indistinguishability of $\mathcal{H}_2$ and $\mathcal{H}_3$ follows from Corollary 4.10. □

Claim 9.14. *For all $i \in [\ell + 2, \ldots, 2\ell + 1]$, $\mathsf{TD}(\mathcal{H}_{i-1}, \mathcal{H}_i) = \mathrm{negl}(\lambda)$.*

Proof. This follows from essentially an identical proof as Claim 9.12. □

Claim 9.15. $\mathsf{TD}(\mathcal{H}_{2\ell+1}, \mathcal{H}_{2\ell+2}) = \mathrm{negl}(\lambda)$.

Proof. This follows from essentially an identical proof as Claim 9.13. □

Claim 9.16. *For all $i \in [2\ell + 3, \ldots, 3\ell + 2]$, $\mathsf{TD}(\mathcal{H}_{i-1}, \mathcal{H}_i) = \mathrm{negl}(\lambda)$.*

Proof. This follows from essentially an identical proof as Claim 9.12. □

□

9.4 Construction of Functional Encryption with Key Revocation

Our construction is a natural generalization of the construction in [GGH+13]. Their construction makes use of a classical public key encryption scheme $(\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{Enc}_{\mathsf{PKE}}, \mathsf{Dec}_{\mathsf{PKE}})$ and statistically simulation sound non-interactive non-interactive zero knowledge proofs (SSS-NIZK). The secret key sk_f consists of an obfuscated program which takes in a SSS-NIZK π and two ciphertexts c_1, c_2, then if π shows that c_1 and c_2 encrypt the same message, it outputs $f(\mathsf{Dec}(c_1))$. Our construction simply implements this functionality as a di$\mathcal{O}$-CD program with a very slight change to allow treating the challenge ciphertext as a differing input.

SSS-NIZK proof systems can be built using statistically-binding commitments and any NIZK proof system [GGH+13]. NIZK proof systems are known from quantum-resistant assumptions [PS19].

Define the language

$$\mathcal{L} = \{(\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_{2,1}, c_{2,2}) : \exists m, r_1, r_2 \text{ s.t. } c_1 = \mathsf{Enc}(\mathsf{pk}_1, m; r_1) \wedge c_2 = \mathsf{Enc}(\mathsf{pk}_2, m \oplus c_{2,2}); r_2)\}$$

This language consists of ciphertexts which encrypt the same message. $c_{2,2}$ acts as a one-time-pad key for the message inside $c_{2,1}$, which will later allow us to argue that some $c^*_{2,1}, c^*_{2,2}$ is hard to find even when we have access to $c^*_{2,1}$ and a one-way function image $g(c^*_{2,2})$. This is necessary for constructing a differing inputs circuit where $(c^*_{2,1}, c^*_{2,2})$ is part of the differing input.

FE Secret Key Functionality

Hardcoded: a function f, a public key pk, and a secret key sk_1 for a classical PKE

Input: NIZK π, ciphertexts c_1 and $(c_{2,1}, c_{2,2})$

1: Parse $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2)$
2: **if** $\mathsf{Verify}_{\mathcal{L}}(\mathsf{crs}, \pi, (\mathsf{pk}_1, \mathsf{pk}_2, c_{2,1}, c_{2,2})) = \top$ **then**
3: Output $f(\mathsf{Dec}(\mathsf{sk}_1, c_1))$.

FE with Key Revocation

$\mathsf{Setup}(1^n)$

1: Sample two classical PKE key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^n)$.
2: Set $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{NIZK}}(1^n)$.
3: Output the public key $\mathsf{pk} = (\mathsf{crs}, \mathcal{L}_{\mathsf{pk}_1,\mathsf{pk}_2}, \mathsf{pk}_1, \mathsf{pk}_2)$ and the master secret key $\mathsf{msk} = \mathsf{sk}_1$.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ Output a di$\mathcal{O}$-CD program for the FE Secret Key Functionality using the parameters $(f, \mathsf{pk}, \mathsf{sk}_1)$.

$\mathsf{Enc}(\mathsf{pk}, m)$

1: Parse $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2)$. Sample $c_{2,2}$ uniformly at random, then compute $c_1 = \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_1, m; r_1)$ and $c_{2,1} = \mathsf{Enc}_{\mathsf{PKE}}(\mathsf{pk}_2, m \oplus c_{2,2}; r_2)$.
2: Compute $\pi \leftarrow \mathsf{Prove}_{\mathcal{L}}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_{2,1}, c_{2,2}), (r_1, r_2))$.
3: Output $(\pi, c_1, c_{2,1}, c_{2,2})$.

$\mathsf{Dec}(\mathsf{sk}_f, c)$

1: Parse $\mathsf{sk}_f = (|t\rangle, \widetilde{C}_{\mathsf{sk}_f})$ and $c = (\pi, c_1, c_{2,1}, c_{2,2})$.
2: Coherently evaluate and output $\widetilde{C}_{\mathsf{sk}_f}(|t\rangle, \pi, c_1, c_{2,1}, c_{2,2})$.

$\mathsf{Del}(\mathsf{sk}_f)$

1: Parse $\mathsf{sk}_f = (|t\rangle, \widetilde{C}_{\mathsf{sk}_f})$.
2: Compute and output the di$\mathcal{O}$-CD deletion proof $\mathsf{cert} \leftarrow \mathsf{Del}_{\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}}(|t\rangle)$.

$\mathsf{Verify}(\mathsf{cert}, \mathsf{vk})$ Output $\mathsf{Verify}_{\mathsf{di}\mathcal{O}\text{-}\mathsf{CD}}(\mathsf{cert}, \mathsf{vk})$.

Theorem 9.17 (Selective FE with CD for Secret Keys)**.** *Assuming the existence of nested differing inputs obfuscation with certified deletion, post-quantum indistinguishability obfuscation, public key encryption,*

and injective one-way functions, there exists functional encryption with (publicly verifiable) selective key revocation.

Remark. We previously constructed nested di$\mathcal{O}$-CD and a classical PKE from post-quantum indistinguishability obfuscation and differing inputs circuits. The existence of injective one-way functions implies the existence of differing inputs circuits.

Proof. Any poly-time adversary makes at most $q = q(n)$ queries over both query phases. For simplicity we assume that they make exactly q queries. We proceed by a hybrid argument where in the first hybrid the challenger encrypts m_0. Then, we gradually change the encryption into an encryption of m_1 using a two-key argument.

- Hyb_0: This is the secret key certified deletion game for functional encryption played with the scheme above using message m_0 in the challenge ciphertext.
- Hyb_1: This hybrid is identical to the previous hybrid, except (crs, π^*) are simulated as

$$(\mathsf{crs}, \mathsf{pk}^*) \leftarrow \mathsf{Sim}(1^n, (\mathsf{pk}_1, \mathsf{pk}_2, c_1^*, c_{2,1}^*, c_{2,2}^*), \mathcal{L})$$

  where the challenge ciphertext is $(\pi^*, c_1^*, c_{2,1}^*, c_{2,2}^*)$. Note that in the selective security game, the challenge ciphertext can be computed before the public key is given to the adversary.
- Hyb_2: This hybrid is identical to the previous hybrid, except the challenge ciphertext is computed using $c_{2,1}^* = \mathsf{Enc}(\mathsf{pk}_2, m_1 \oplus c_{2,2}^*)$. Recall that the NIZK π^* is simulated according to $(c_1^*, c_{2,1}^*, c_{2,2}^*)$.
- $\mathsf{Hyb}_{3,i}$ for $i = 1, \ldots, q$: In this series of hybrids, we change the form of the functional secret keys sk_f. In $\mathsf{Hyb}_{3,i}$, the first i functional secret keys requested are computed as obfuscations of the Hyb_3 program. The remaining $i+1$ to q keys are generated as in Hyb_2. $\mathsf{Hyb}_{3,0}$ is exactly Hyb_2. Let g be an injective one-way function.

Hyb_3 Program

Hardcoded: a function f, a public key pk, and secret key sk_1 for a classical PKE, the message m_1, the ciphertexts c_1^*, $c_{2,1}^*$, and the value $g(c_{2,2}^*)$
Input: NIZK π, ciphertexts c_1 and $(c_{2,1}, c_{2,2})$

1: Parse $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2)$
2: **if** $\mathsf{Verify}_{\mathcal{L}}(\mathsf{crs}, \pi, (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_{2,1}, c_{2,2})) = \top$ **then**
3: **if** $c_1 = c_1^*$, $c_{2,1} = c_{2,1}^*$ and $g(c_{2,2}) = g(c_{2,2}^*)$ **then**
4: Output $f(m_1)$.
5: **else**
6: Output $f(\mathsf{Dec}(\mathsf{sk}_1, c_1))$.

- $\mathsf{Hyb}_{4,i}$ for $i = 1, \ldots, q$: In this series of hybrids, we change the form of the functional secret keys sk_f. In $\mathsf{Hyb}_{4,i}$, the first i functional secret keys requested are computed as obfuscations of the Hyb_4 program. The remaining $i+1$ to q keys are generated as in $\mathsf{Hyb}_{3,q}$. $\mathsf{Hyb}_{4,0}$ is exactly $\mathsf{Hyb}_{3,q}$.

Hyb_4 Program

Hardcoded: a function f, a public key pk, and secret keys $\mathsf{sk}_1, \mathsf{sk}_2$ for a classical PKE

Input: NIZK π, ciphertexts c_1 and $(c_{2,1}, c_{2,2})$

1: Parse $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2)$
2: **if** $\mathsf{Verify}_{\mathcal{L}}(\mathsf{crs}, \pi, (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_{2,1}, c_{2,2})) = \top$ **then**
3: Output $f(\mathsf{Dec}(\mathsf{sk}_2, c_{2,1}) + c_{2,2})$.

- Hyb_5: This hybrid is identical to the previous hybrid, except the challenge ciphertext is computed using $c_1^* = \mathsf{Enc}(\mathsf{pk}_1, m_1)$.
- $\mathsf{Hyb}_{6,i}$ for $i = 1, \ldots, q$: In this series of hybrids, we change the form of the functional secret keys sk_f. In $\mathsf{Hyb}_{6,i}$, the first i functional secret keys requested are computed as obfuscations of the FE secret key program. The remaining $i+1$ to q keys are generated as in Hyb_5. $\mathsf{Hyb}_{6,0}$ is exactly Hyb_5.
- Hyb_6: This hybrid is identical to the previous hybrid, except the crs and NIZK proof π^* in the challenge ciphertext are generated honestly. This hybrid corresponds to the secret key certified deletion game using message m_1 in the challenge ciphertext.

Claim 9.18. *If the SSS-NIZK system is computationally zero knowledge, then* Hyb_0 *is computationally indistinguishable from* Hyb_1.

Proof. This is immediate from the zero knowledge property, since the rest of the game can be simulated by a reduction which internally generates the PKE keys. □

Claim 9.19. *If the classical PKE scheme is semantically secure,* Hyb_1 *is computationally indistinguishable from* Hyb_2.

Proof. This is immediate from the semantic security of the classical PKE scheme, since the rest of the game can be simulated by a reduction which internally generates just $(\mathsf{pk}_1, \mathsf{sk}_1)$. □

Claim 9.20. *If the* di$\mathcal{O}$-CD *scheme has nested differing inputs certified deletion, the classical PKE is semantically secure, and if g is an injective one-way function,* $\mathsf{Hyb}_{3,i}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{3,i+1}$. *Note that* $\mathsf{Hyb}_2 = \mathsf{Hyb}_{3,0}$.

Proof. We can rewrite the FE secret key program and the Hyb_3 program in nested form as follows:

FE Secret Key Outer Program

Hardcoded: a public key pk and an inner program C

Input: NIZK π, ciphertexts c_1 and $(c_{2,1}, c_{2,2})$

1: Parse $\mathsf{pk} = (\mathsf{crs}, \mathsf{pk}_1, \mathsf{pk}_2)$
2: **if** $\mathsf{Verify}_{\mathcal{L}}(\mathsf{crs}, \pi, (\mathsf{pk}_1, \mathsf{pk}_2, c_1, c_{2,1}, c_{2,2})) = \top$ **then**
3: Output $C(c_1, c_{2,1}, c_{2,2})$

FE Secret Key Inner Program

Hardcoded: a function f and secret key sk_1 for a classical PKE

Input: ciphertexts c_1 and $(c_{2,1}, c_{2,2})$

1: Output $f(\mathsf{Dec}(\mathsf{sk}_1, c_1))$.

Hyb_3 Inner Program

Hardcoded: a function f, secret key sk_1 for a classical PKE, the message m_1, the ciphertexts c_1^* and $c_{2,1}^*$, and the one-way function image $g(c_{2,2}^*)$

Input: ciphertexts c_1 and $(c_{2,1}, c_{2,2})$

1: **if** $c_1 = c_1^*$, $c_{2,1} = c_{2,1}^*$, and $g(c_{2,2}) = g(c_{2,2}^*)$ **then**
2: Output $f(m_1)$.
3: **else**
4: Output $f(\mathsf{Dec}(\mathsf{sk}_1, c_1))$.

The FE secret key program is obtained by instantiating the FE secret key outer program with the FE secret key inner program. The Hyb_3 program is obtained by instantiating it with the Hyb_3 inner program. Using the injectiveness of the one-way function, it is easy to verify that the two inner programs differ in at most one input: $(c_1^*, c_{2,1}^*, c_{2,2}^*)$. To show that the inner programs are differing inputs circuits, we need to show that $(c_1^*, c_{2,1}^*, c_{2,2}^*)$ is hard to find given their descriptions and the auxiliary information the adversary has when the challenger creates the obfuscated circuits, which is the message pair (m_0, m_1) and the description of the FE secret key outer program without the hardcoded inner program C. Note that this auxiliary input is entirely classical, as required by our definition of di$\mathcal{O}$-CD. It suffices to show that the adversary cannot even find $c_{2,2}^*$.

Claim 9.21. *Assuming the properties from claim 9.20, for any (m_0, m_1) and any function f, we have*

$$\Pr\left[\mathcal{A}(C_0, C_1, \mathsf{aux}) = c_{2,2}^* : \begin{array}{c}(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{KeyGen}_{\mathsf{PKE}}(1^n), \\ c_{2,2}^* \leftarrow \{0,1\}^n, \\ c_1^* \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m_0), c_{2,1}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_2, m_1 \oplus c_{2,2}^*), \\ C_0 = \mathsf{FEInner}(f, \mathsf{sk}_1), \\ C_1 = \mathsf{Hyb}_3\mathsf{Inner}(f, \mathsf{sk}_1, m_1, c_1^*, c_{2,1}^*, g(c_{2,2}^*)) \\ \mathsf{aux} = (\mathsf{pk}_1, \mathsf{pk}_2, m_0, m_1, f)\end{array}\right] = \mathrm{negl}(\lambda)$$

Proof. Consider a hybrid where $c_{2,1}^*$ is generated as an encryption of 0 instead of being an encryption of $m_1 \oplus c_{2,2}^*$. Since C_0, C_1, and aux can be generated given $c_{2,1}^*$ and pk_2, without knowledge of sk_2, this hybrid is indistinguishable from the experiment above due to the semantic security of the PKE. Note that in this hybrid, $c_{2,2}^*$ is independent of m_0 and m_1 even given $c_{2,1}^*$. Furthermore, C_0 and C_1 can be computed just using $g(c_{2,2}^*)$ instead of $c_{2,2}^*$. Since $c_{2,2}^*$ is uniformly random and independent of the auxiliary information, by the one-way property of g we have $\Pr[\mathcal{A}(C_0, C_1, \mathsf{aux}) = c_{2,2}^*] = \mathrm{negl}(\lambda)$ in this hybrid. Since this hybrid is indistinguishable from the original experiment, we have the claim. □

Therefore the nested differing inputs certified deletion property of the di$\mathcal{O}$-CD scheme ensures

that, for any m_0, m_1 ,and f_i, the deletion game played with the FE secret key program for sk_{f_i} is indistinguishable from the one played with the Hyb_3 program for sk_{f_i}, i.e. $\mathsf{Hyb}_{3,i}$ is computationally indistinguishable from $\mathsf{Hyb}_{3,i+1}$. □

Claim 9.22. *If the* di$\mathcal{O}$-CD *scheme is an indistinguishability obfuscator and the NIZK is statistically simulation sound,* $\mathsf{Hyb}_{4,i}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{4,i+1}$. *Note that* $\mathsf{Hyb}_{3,q} = \mathsf{Hyb}_{4,0}$.

Proof. It suffices to show that the Hyb_3 program is functionally equivalent to the Hyb_4 program, since then indistinguishability obfuscation immediately implies the indistinguishability of the two hybrids. Consider any input $(\pi, c_1, c_{2,1}, c_{2,2})$. There are three cases:

- π is rejecting. In this case, both programs output $\perp$.
- π is accepting and $(c_1, c_{2,1}, c_{2,2}) = (c_1^*, c_{2,1}^*, c_{2,2}^*)$. In this case, both programs output $f(\mathsf{Dec}(\mathsf{sk}_2, c_{2,1}) \oplus c_{2,2}) = f(m_1)$.
- π is accepting and $(c_1, c_{2,1}, c_{2,2}) \neq (c_1^*, c_{2,1}^*, c_{2,2}^*)$. In this case, due to the statistical simulation soundness of the NIZK, $\mathsf{Dec}(\mathsf{sk}_1, c_1) = \mathsf{Dec}(\mathsf{sk}_2, c_{2,1}) \oplus c_{2,2}$. Therefore both programs have the same output $f(\mathsf{Dec}(\mathsf{sk}_1, c_1))$.

□

Claim 9.23. *If the classical PKE scheme is semantically secure,* Hyb_5 *is computationally indistinguishable from* $\mathsf{Hyb}_{4,q}$.

Proof. This is immediate from the semantic security of the classical PKE scheme, since the rest of the game can be simulated by a reduction which internally generates $(\mathsf{pk}_2, \mathsf{sk}_2)$. □

Claim 9.24. *If the* di$\mathcal{O}$-CD *scheme is an indistinguishability obfuscator and the NIZK is statistically simulation sound,* $\mathsf{Hyb}_{6,i}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{6,i+1}$. *Note that* $\mathsf{Hyb}_5 = \mathsf{Hyb}_{6,0}$.

Proof. It suffices to show that the Hyb_4 program is functionally equivalent to the FE secret key program, since then indistinguishability obfuscation immediately implies the indistinguishability of the two hybrids. Consider any input $(\pi, c_1, c_{2,1}, c_{2,2})$. There are two cases:

- π is rejecting. In this case, both programs output $\perp$.
- π is accepting. In this case, due to the statistical simulation soundness of the NIZK, $\mathsf{Dec}(\mathsf{sk}_1, c_1) = \mathsf{Dec}(\mathsf{sk}_2, c_{2,1}) \oplus c_{2,2}$. This holds even for $(c_1, c_{2,1}, c_{2,2}) = (c_1^*, c_{2,1}^*, c_{2,2}^*)$. Therefore both programs have the same output $f(\mathsf{Dec}(\mathsf{sk}_2, c_{2,1}) \oplus c_{2,2})$.

□

Claim 9.25. *If the NIZK is computationally zero knowledge,* $\mathsf{Hyb}_{6,q}$ *is computationally indistinguishable from* Hyb_7.

Proof. This is immediate from the zero knowledge property, since the rest of the game can be simulated by a reduction which internally generates the PKE keys. □

Therefore the game when played when the challenge ciphertext is an encryption of m_0 is indistinguishable from the game when the challenge ciphertext is an encryption of m_1. □

10 Acknowledgments

We thank Sanjam Garg for collaboration and valuable contributions during the earlier stages of this research.

D.K. was supported by DARPA SIEVE, AFOSR, NSF CAREER CNS-2238718 and NSF CNS-2247727. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024, and by the Air Force Office of Scientific Research under award number FA9550-23-1-0543.

G.M. was supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation).

J.R was supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 229–242. IEEE Computer Society, 2009.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. https://eprint.iacr.org/2013/689.

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 41–60, New York, NY, USA, 2012. Association for Computing Machinery.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.

[AK21] Prabhanjan Ananth and Fatih Kaleoglu. Unclonable encryption, revisited. LNCS, pages 299–329. Springer, Heidelberg, 2021.

[AK22] Prabhanjan Ananth and Fatih Kaleoglu. A note on copy-protection from random oracles. Cryptology ePrint Archive, Paper 2022/1109, 2022. https://eprint.iacr.org/2022/1109.

[AKL+22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. On the feasibility of unclonable encryption, and more. CRYPTO, 2022.

[AKL23] Prabhanjan Ananth, Fatih Kaleoglu, and Qipeng Liu. Cloning games: A general framework for unclonable primitives. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 66–98, Cham, 2023. Springer Nature Switzerland.

[AKN+23] Shweta Agrawal, Fuyuki Kitagawa, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Public key encryption with secure key leasing. Cryptology ePrint Archive, Paper 2023/264, 2023. https://eprint.iacr.org/2023/264.

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. LNCS, pages 501–530. Springer, Heidelberg, 2021.

[APV23] Prabhanjan Ananth, Alexander Poremba, and Vinod Vaikuntanathan. Revocable cryptography from learning with errors. Cryptology ePrint Archive, Paper 2023/325, 2023. https://eprint.iacr.org/2023/325.

[BB84] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, pages 175–179, 1984.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

[BDGM22] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and Pairings Are Not Necessary for IO: Circular-Secure LWE Suffices. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 544–574. Springer, Heidelberg, November 2018.

[BI20] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. LNCS, pages 92–122. Springer, Heidelberg, March 2020.

[BJL+21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions. LNCS, pages 90–120. Springer, Heidelberg, 2021.

[BK23] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 192–223, Cham, 2023. Springer Nature Switzerland.

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 494–524. Springer, Heidelberg, March 2017.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 474–502. Springer, Heidelberg, January 2016.

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 792–821. Springer, Heidelberg, May 2016.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Snargs for $\mathcal{P}$ from LWE. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 68–79. IEEE, 2021.

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. LNCS, pages 556–584. Springer, Heidelberg, 2021.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments: Breaking the quantum rewinding barrier. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 49–58. IEEE, 2021.

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018.

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. LNCS, pages 256–287. Springer, Heidelberg, 2021.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, August 2014.

[GJO16] Vipul Goyal, Aayush Jain, and Adam O'Neill. Multi-input functional encryption with unbounded-message security. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 531–556. Springer, Heidelberg, December 2016.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 736–749, New York, NY, USA, 2021. Association for Computing Machinery.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[HKM+23] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption, and more. Cryptology ePrint Archive, Paper 2023/236, 2023. https://eprint.iacr.org/2023/236.

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. LNCS, pages 606–636. Springer, Heidelberg, 2021.

[HMNY22a] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting functional encryption. *CoRR*, abs/2207.13878, 2022.

[HMNY22b] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting zero-knowledge proof for QMA, 2022. Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, Santa Barbara, CA, USA.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 419–428, New York, NY, USA, 2015. Association for Computing Machinery.

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Functional encryption with secure key leasing. In *Advances in Cryptology - ASIACRYPT 2022*. Springer Cham, 2022.

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. LNCS, pages 31–61. Springer, Heidelberg, 2021.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

[KTZ13] Jonathan Katz, Aishwarya Thiruvengadam, and Hong-Sheng Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 14–31. Springer, Heidelberg, February / March 2013.

[NY01] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. *ACM Symposium on Theory of Computing*, 03 2001.

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 90:1–90:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Shm22] Omri Shmueli. Semi-quantum tokenized signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 296–319. Springer, 2022.

[Unr14] Dominique Unruh. Revocable quantum timed-release encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 129–146. Springer, Heidelberg, May 2014.

[VZ21] Thomas Vidick and Tina Zhang. Classical proofs of quantum knowledge. LNCS, pages 630–660. Springer, Heidelberg, 2021.

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15:78–88, 1983.

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. LNCS, pages 127–156. Springer, Heidelberg, 2021.

[WZ82] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, Oct 1982.

[Zha19] Mark Zhandry. Quantum lightning never strikes the same state twice. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 408–438. Springer, Heidelberg, May 2019.

A Auxiliary Lemmas from Section 9

We prove the following lemma and its corollary, which will assist in our proof of FE with certified deletion. This is a variant of (and follows a proof structure that is similar to the proof of) subspace-hiding obfuscation [Zha19], but tailored to functional encryption.

Lemma A.1 (Subspace-Hiding for Multi-input Functional Encryption)**.** *Any multi-input functional encryption scheme* $(\mathsf{MIFE\text{-}Setup}, \mathsf{MIFE\text{-}KeyGen}, \mathsf{MIFE\text{-}Enc}_1, \mathsf{MIFE\text{-}Enc}_2, \mathsf{MIFE\text{-}Dec})$ *of arity two (Definition 9.4) satisfies the following.*

$$\Pr\left[b'=b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ (\mathsf{pp},\mathsf{msk}) \leftarrow \mathsf{MIFE\text{-}Setup}(1^\lambda) \\ M \leftarrow \mathcal{A}^{\mathsf{MIFE\text{-}KeyGen}(\mathsf{msk},g(\cdot))}(\mathsf{pp}) \\ \mathsf{ct} \leftarrow \mathsf{MIFE\text{-}Enc}_2(S,T,\mathsf{u},\mathsf{v},M,0,0) \text{ if } b=0 \\ \mathsf{ct} \leftarrow \mathsf{MIFE\text{-}Enc}_2(S,0,0,\mathsf{v},M,1,0) \text{ if } b=1 \\ b' \leftarrow \mathcal{A}^{\mathsf{MIFE\text{-}KeyGen}(\mathsf{msk},g(\cdot))}(\mathsf{pp},\mathsf{sk}_f,\mathsf{ct},S,\mathsf{v}) \end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is over the randomness of sampling $S < T < \mathbb{F}_2^n$ *such that* $\dim(S) = d$ *and* $\dim(T) = d+1$ *where* $d \in [n/2, 3n/4]$ *for* $n = 4\lambda$*,* $\mathsf{v},\mathsf{w} \leftarrow \mathsf{co}(S) \times \mathsf{co}(S^\perp)$*, and letting* u *be the coset of* T *that* v *belongs to. Moreover for any function* f *the related function* $g(f)$ *is defined as follows. First, obtain input* $(\mathsf{t},(S,T,\mathsf{u},\mathsf{v},M,c,\mathsf{td}))$ *and parse* $\mathsf{td} = (c_1,i,y)$*. Then,*

- *If* $c_1 = 0$*, do:*
  - *If* $c=0$*, if* $\mathsf{t} \notin T + \mathsf{u}$*, then abort and output* $\perp$*, and otherwise output* $f(M)$*.*
  - *If* $c=1$*, if* $\mathsf{t} \notin S + \mathsf{v}$*, then abort and output* $\perp$*, and otherwise output* $f(M)$*.*

  (We note that the above "If $c_1 = 0$*" is the only branch that is invoked in the real experiment, the following if statements only support the proof.)*
- *If* $c_1 = 1$*, do:*
  - *If* $c=0$*, if* $\mathsf{t} - \mathsf{u} \notin T$*, then abort and output* $\perp$*, and otherwise output* $f(M)$*.*
  - *If* $c=1$*, if* $\mathsf{t} - \mathsf{v} \notin S$*, then abort and output* $\perp$*, and otherwise output* $f(M)$*.*
- *If* $c_1 = 2$*, do:*
  - *If* $c=0$*, then compute* B *to be the* $(n-d)\times n$ *matrix whose rows are a basis for* $S^\perp$*, the space orthogonal to* S*. Then if* $\mathsf{B}\cdot(\mathsf{t}-\mathsf{v}) \neq 0$ *and* $\widehat{G}_y(\mathsf{B}\cdot(\mathsf{t}-\mathsf{v})) \neq 1$*, then abort and output* $\perp$ *and otherwise output* $f(M)$*. Here* $\widehat{G}_y$ *is a program that outputs* 1 *on input* x *where* $f(x) = y$*, and otherwise outputs* 0*.*
  - *If* $c=1$*, then compute* B *to be the* $(n-d)\times n$ *matrix whose rows are a basis for* $S^\perp$*, the space orthogonal to* S*. Then if* $\mathsf{B}\cdot(\mathsf{t}-\mathsf{v}) \neq 0$ *and* $\widehat{G}(\mathsf{B}\cdot(\mathsf{t}-\mathsf{v})) \neq 1$*, then abort and output* $\perp$ *and otherwise output* $f(M)$*. Here* $\widehat{G}$ *is the all-zeroes program.*
- *If* $c_1 = 3$*, do:*

– *If* $\mathsf{t} \leq i$*, then compute* B *to be the* $(n-d) \times n$ *matrix whose rows are a basis for* $S^{\perp}$*, the space orthogonal to* S*. Then if* $\mathsf{B} \cdot (\mathsf{t} - \mathsf{v}) \neq 0$ *and* $\widehat{G}_y(\mathsf{B} \cdot (\mathsf{t} - \mathsf{v})) \neq 1$*, then abort and output* $\perp$ *and otherwise output* $f(M)$*. Here* $\widehat{G}_y$ *is a program that outputs* 1 *on input* x *where* $f(x) = y$*, and otherwise outputs* 0*.*

– *If* $\mathsf{t} < i$*, then compute* B *to be the* $(n-d) \times n$ *matrix whose rows are a basis for* $S^{\perp}$*, the space orthogonal to* S*. Then if* $\mathsf{B} \cdot (\mathsf{t} - \mathsf{v}) \neq 0$ *and* $\widehat{G}(\mathsf{B} \cdot (\mathsf{t} - \mathsf{v})) \neq 1$*, then abort and output* $\perp$ *and otherwise output* $f(M)$*. Here* $\widehat{G}$ *is the all-zeroes program.*

Proof. We consider the following sequence of hybrids.

- $\mathcal{H}_0$: ct is an encryption of $(0, T, \mathsf{u}, 0, M, 0, \mathsf{td})$ where $\mathsf{td} = (0, 0, 0)$.
- $\mathcal{H}_1$: ct is an encryption of $(S, T, \mathsf{u}, \mathsf{v}, M, 0, \mathsf{td})$ where $\mathsf{td} = (1, 0, 0)$.
- $\mathcal{H}_2$: ct is an encryption of $M_2 = (S, 0, \mathsf{u}, \mathsf{v}, M, 0, \mathsf{td})$ for $\mathsf{td} = (2, 0, y)$, $y = f(x)$ for $x \leftarrow \{0,1\}^{n-d}, x \neq 0$.
- $\mathcal{H}_3$: ct is an encryption of $M_3 = (S, 0, \mathsf{u}, \mathsf{v}, M, 1, \mathsf{td})$ where $\mathsf{td} = (2, 0, 0)$.
- $\mathcal{H}_4$: ct is an encryption of $(S, 0, \mathsf{u}, \mathsf{v}, M, 1, \mathsf{td})$ where $\mathsf{td} = (1, 0, 0)$.
- $\mathcal{H}_5$: ct is an encryption of $(S, 0, 0, \mathsf{v}, M, 1, \mathsf{td})$ where $\mathsf{td} = (0, 0, 0)$.

Indistinguishability between hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$, between hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$, and between hybrids $\mathcal{H}_4$ and $\mathcal{H}_5$ follows straightforwardly from the security of MIFE (Definition 9.4).

Indistinguishability between hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$ also follows from the security of MIFE as follows. Consider sampling $x^* \leftarrow \{0,1\}^{n-d}$ such that $x^* \neq 0$, then computing $y = f(x^*)$, and T as the subspace of vectors t' such that $B \cdot \mathsf{t}'$ is in the span of x^*. For challenge messages $M_0 = (S, T, \mathsf{u}, \mathsf{v}, M, 0, \mathsf{td})$ for $\mathsf{td} = (2, 0, y)$ and $M_1 = (S, T, \mathsf{u}, \mathsf{v}, M, 1, \mathsf{td})$ for $\mathsf{td} = (2, 0, 0)$, indistinguishability follows by security of MIFE since $g_f(\cdot, M_0) = g_f(\cdot, M_1)$.

Finally, indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_3$ follows by an argument inspired by [BCP14] for the setting of extractability/differing-inputs obfuscation. In more detail, suppose these hybrids are distinguishable with non-negligible advantage $\epsilon = \epsilon(n)$. Then we will build an inverter I that inverts the one-way function on a random y with probability $\frac{\epsilon}{2}$, leading us to a contradiction. The inverter I given y aims to find x^* such that $f(x^*) = y$. The inverter proceeds as follows.

1. Set $i = 2^{n-d-1}$.
2. While $i > 0$, repeat:
   (a) Define $\mathcal{H}_{\mathsf{mid}}$ where ct is generated as an encryption of $M_{\mathsf{mid}} = (S, T, \mathsf{u}, \mathsf{v}, M, 0, \mathsf{td})$ for $\mathsf{td} = (3, i, y)$.
   (b) Estimate the advantage ϵ_L of the adversary between $\mathcal{H}_2$ and $\mathcal{H}_{\mathsf{mid}}$ with estimation error bounded by $\frac{\epsilon}{2n}$ w.h.p. (this requires $O(\frac{n^2}{\epsilon^2})$ samples). If $\epsilon_L > \epsilon \cdot (\frac{n-1}{n})$, set $x^*[1] = 0$ and rename $\mathcal{H}_{\mathsf{mid}}$ to $\mathcal{H}_3$. Otherwise set $x^*[1] = 1$ and rename $\mathcal{H}_{\mathsf{mid}}$ to $\mathcal{H}_2$.
3. Output x^*.

By a union bound, error in estimating any bit of x^* is bounded by $\frac{\epsilon}{2n} \cdot (n-d) + \mathrm{negl}(n) \leq \frac{\epsilon}{2}$. Therefore, the inverter given y runs in polynomial time and successfully outputs the inverse of y with probability $\frac{\epsilon}{2}$. This is a contradiction as desired. □

Applying this lemma repeatedly immediately yields the following corollary, which is the full version of Corollary 9.9, where the dimensions between subspaces S and T does not differ by 1, but by upto $\frac{n}{4}$.

Corollary A.2. *(Subspace-Hiding for Multi-input Functional Encryption) Any multi-input functional encryption scheme* $(\mathsf{MIFE\text{-}Setup}, \mathsf{MIFE\text{-}KeyGen}, \mathsf{MIFE\text{-}Enc}_1, \mathsf{MIFE\text{-}Enc}_2, \mathsf{MIFE\text{-}Dec})$ *of arity two (Definition 9.4) satisfies the following.*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{MIFE\text{-}Setup}(1^\lambda) \\ M \leftarrow \mathcal{A}^{\mathsf{MIFE\text{-}KeyGen}(\mathsf{msk}, g(\cdot))}(\mathsf{pp}) \\ \mathsf{ct} \leftarrow \mathsf{MIFE\text{-}Enc}_2(S, T, \mathsf{u}, \mathsf{v}, M, 0, 0) \text{ if } b = 0 \\ \mathsf{ct} \leftarrow \mathsf{MIFE\text{-}Enc}_2(S, 0, 0, \mathsf{v}, M, 1, 0) \text{ if } b = 1 \\ b' \leftarrow \mathcal{A}^{\mathsf{MIFE\text{-}KeyGen}(\mathsf{msk}, g(\cdot))}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}, S, \mathsf{v}) \end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is over the randomness of sampling $S < T < \mathbb{F}_2^n$ such that $\dim(S) \leq \dim(T)$ and both dimensions are in $[n/2, 3n/4]$ for $n = 4\lambda$, $\mathsf{v}, \mathsf{w} \leftarrow \mathsf{co}(S) \times \mathsf{co}(S^\perp)$, and letting u be the coset of T that v belongs to. Moreover for any function f the related function $g(f)$ is defined as follows. First, obtain input $(\mathsf{t}, (S, T, \mathsf{u}, \mathsf{v}, M, c, \mathsf{td}))$ and parse $\mathsf{td} = (c_1, i, y)$. Then,

- *If $c_1 = 0$, do:*
  - *If $c = 0$, if $\mathsf{t} \notin T + \mathsf{u}$, then abort and output $\perp$, and otherwise output $f(M)$.*
  - *If $c = 1$, if $\mathsf{t} \notin S + \mathsf{v}$, then abort and output $\perp$, and otherwise output $f(M)$.*
- *If $c_1 = 1$, do:*
  - *If $c = 0$, if $\mathsf{t} - \mathsf{u} \notin T$, then abort and output $\perp$, and otherwise output $f(M)$.*
  - *If $c = 1$, if $\mathsf{t} - \mathsf{v} \notin S$, then abort and output $\perp$, and otherwise output $f(M)$.*
- *If $c_1 = 2$, do:*
  - *If $c = 0$, then compute B to be the $(n-d) \times n$ matrix whose rows are a basis for $S^\perp$, the space orthogonal to S. Then if $\mathsf{B} \cdot (\mathsf{t} - \mathsf{v}) \neq 0$ and $\widehat{G}_y(\mathsf{B} \cdot (\mathsf{t} - \mathsf{v})) \neq 1$, then abort and output $\perp$ and otherwise output $f(M)$. Here $\widehat{G}_y$ is a program that outputs 1 on input x where $f(x) = y$, and otherwise outputs 0.*
  - *If $c = 1$, then compute B to be the $(n-d) \times n$ matrix whose rows are a basis for $S^\perp$, the space orthogonal to S. Then if $\mathsf{B} \cdot (\mathsf{t} - \mathsf{v}) \neq 0$ and $\widehat{G}(\mathsf{B} \cdot (\mathsf{t} - \mathsf{v})) \neq 1$, then abort and output $\perp$ and otherwise output $f(M)$. Here $\widehat{G}$ is the all-zeroes program.*
- *If $c_1 = 3$, do:*
  - *If $\mathsf{t} \leq i$, then compute B to be the $(n-d) \times n$ matrix whose rows are a basis for $S^\perp$, the space orthogonal to S. Then if $\mathsf{B} \cdot (\mathsf{t} - \mathsf{v}) \neq 0$ and $\widehat{G}_y(\mathsf{B} \cdot (\mathsf{t} - \mathsf{v})) \neq 1$, then abort and output $\perp$ and otherwise output $f(M)$. Here $\widehat{G}_y$ is a program that outputs 1 on input x where $f(x) = y$, and otherwise outputs 0.*

– *If* $\mathsf{t} < i$*, then compute* B *to be the* $(n-d) \times n$ *matrix whose rows are a basis for* $S^{\perp}$*, the space orthogonal to* S*. Then if* $\mathsf{B} \cdot (\mathsf{t} - \mathsf{v}) \neq 0$ *and* $\widehat{G}(\mathsf{B} \cdot (\mathsf{t} - \mathsf{v})) \neq 1$*, then abort and output* $\perp$ *and otherwise output* $f(M)$*. Here* $\widehat{G}$ *is the all-zeroes program.*