Quantum Approaches to Verifiable Deletion

Justin Raizes

CMU-CS-25-107
May 2025

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Vipul Goyal (Chair)
Aayush Jain
Elaine Shi
Giulio Malavolta (Bocconi University)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science.

This research was sponsored by JP Morgan, PNC Center for Financial Services, the National Science Foundation under award number CNS-1916939, the Department of Energy under award number DEFE0031770, and the Defense Advanced Research Projects Agency under award number HR0011-20-20025.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Quantum Cryptography, Certified Deletion, Obfuscation, Secret Sharing, Signatures, Zero Knowledge

To those who inspire others to learn.

Abstract

Intriguingly, the laws of physics allow protocols executed by quantum computers to realize security guarantees that are *impossible* for classical computers. One of these new possibilities is the ability to temporarily send ciphertexts to a user, then later verify that the encoded message has been destroyed in an information-theoretic sense.

Broadbent and Islam (TQC) introduced this notion, called certified deletion, in the context of encryption. However, there are many more scenarios in which verifiable deletion is useful. For example, a software rental company might want to temporarily send their software to a user, then request that the user destroys the copy at the end of the rental period.

In this thesis, we expand the realm of certified deletion to cryptographic primitives beyond just encryption. We define and construct the following objects with certified deletion:

- **Obfuscation with Certified Deletion** allows a company to lend a program to a user, allowing them to evaluate it as they wish. Then, when the user no longer wish to rent the program, they can destroy it and prove to the issuer that they are no longer able to evaluate the program. Obfuscation with certified deletion also enables several new applications such as certifiably deletable secret keys.
- **Secret Sharing with Certified Deletion** allows a user to distribute shares of a secret to several parties. In the event of a data breach, the user can request that the affected party deletes their shares, rendering them useless for stealing the secret.
- **Signatures with Certified Deniability** allow a signer to endorse a statement in a single message. After the receiver has verified the signature, they can destroy it and prove to the signer that they are no longer able to provide convincing evidence - of any kind - that the signer endorsed this statement. We also show how to construct the related primitive of NIZKs with certified deniability.

  Certified deniability is a new, more comprehensive paradigm for certified deletion that rules out additional attacks not explicitly considered by prior definitions.

To build these primitives, we develop new techniques for verifying the deletion of information while still allowing access to the information under appropriate conditions.

Acknowledgments

Getting to this point has been a long journey filled with people who supported me along the way. I am lucky to have so many friends, family, and mentor figures.

Thank you to my advisor, Vipul. You were patient with me as I transitioned into crypto, introduced me to many members of the crypto community, and helped me find new and exciting topics to study.

Thank you to the members of my committee, who were there for me at the end of this path.

Thank you to my friends from UCSC. Yash, Evan, Yitzhar, Alicia, Matthew Gray, Matthew Rhea, Mace, and many more, you filled my time at UCSC with joy and made me happy to be there. Your friendship has followed me beyond UCSC and been an amazing support over these years. A special thank you to Matthew Gray, whose excitement about math and computer science is not only infectious, but also recursive. Our many talks and the classes we taught together confirmed that embarking on the long road to a Ph.D. was the right choice for me. Speaking of those classes, thank you to the students who believed that Matthew and I had enough interesting things to say that it was worth showing up at 8 A.M. to listen to us rant about computer science.

Thank you to my mentors during my time at UCSC, who helped me find opportunities to progress along this path. To Darrell, Ethan, and Sesh, who enabled my teaching- and research-related shenanigans at UCSC. To Tom, who hosted me at Sandia for so many years and encouraged me to explore new topics freely.

Thank you to the mentors who helped me really kick off my PhD. My productivity before and after I started working with post-docs is as night and day. So thank you to those post-docs: Pratik, Chen-Da, and João. Thank you to Giulio for hosting me in Germany after COVID and helping me get used to working in person again.

Thank you to my many collaborators. You all taught me so much as we worked together. I have no doubt that my list of publications would not be in its current state without you.

Thank you to the student community, both within crypto and without. My friends at CMU made my journey much more enjoyable. Thank you to my board game buddies: Sara, Sol, Isaac, Ray, Josh, and everyone else who showed up to play board games (somewhat) weekly. Thank you as well to those who didn't play board games with me, but who I still looked forward to hanging out with every time. Thank you to Mik, who was an amazing roommate for almost my entire Ph.D. and was a great friend for its entire span. Anup, Arjun, and Eitan, you lived with me more briefly but were still amazing friends. Thank you for your company. Thank you to my officemates Hugo, Long, Justin, Sid, Yonghao, Mingjie, and our newest addition Runming for making each workday more companionable. The interns at NTT played a similar role during my internship there, except instead of the gym it was just one more round of Hanabi (even if we'd just finished the "one" round). Thank you for injecting some excitement into those summers. Thank you to the many crypto grad students whose Ph.D.s overlapped with mine. I always looked forward to chatting with you all at conferences about the latest research trends and brainstorming crazy ideas, even if they didn't work out most of the time.

Thank you to my family, who was there from the very beginning and who always believed in me.

Contents

Chapter 1

Introduction

It is oft-repeated advice to be careful of what one posts on the internet, because nothing ever disappears once is it out there. This is certainly true with classical computers. Anyone could have screenshotted, downloaded, or otherwise copied the information.

However, quantum mechanics offers a way around this limitation. The no-cloning theorem [WZ82] explicitly disallows copying general quantum states. Broadbent and Islam [BI20a] leveraged this idea to verifiably delete messages from ciphertexts, which they called "encryption with certified deletion". In encryption with certified deletion, a sender encrypts a message in a quantum state $|\psi_m\rangle$ and sends it to a receiver. The receiver can later destructively measure $|\psi_m\rangle$ to produce a certificate. If the certificate is valid - as determined by the sender - then the receiver cannot recover any information about the message even if they steal the decryption key.

Beyond just encryption, there are other applications in which verifiable deletion is desirable. For example, deletable programs could be useful in software rental; the company could send a copy to a user, then ask the user to delete it at the end of the rental period. As another example, a customer whose data is split across multiple servers could ask one of them to delete the stored data in the event of a data breach, preventing attackers from accumulating pieces of it over time. The purpose of this thesis is to shed light on the question

What else does quantum mechanics allow to be certifiably deleted?

In pursuit of answers, I investigate how to define certified deletion for new cryptographic objects, how to construct objects satisfying those definitions, and what the limits of certified deletion are. This thesis includes the following answers:

- It is possible to delete **programs**. A company may prepare an encoded program and send it to a user, who may evaluate it as they wish. After the company sees a valid deletion certificate, they know that the user can no longer evaluate the program.
- It is possible to delete **secret shares**. A user may split a classical secret into n quantum shares, where k of the shares may be used to reconstruct the secret. However, verifiably deleted shares cannot contribute to reconstruction.
- It is possible to delete **all evidence of a signature**. A sender may sign a message m, so that anyone may verify they said it. After the sender sees a valid deletion certificate, they know that no one can provide any evidence that any message was previously signed. Similarly, it is possible to delete **all evidence of a non-interactive zero-knowledge proof**.

1.1 Results

Obfuscation with Certified Deletion. Classically, obfuscation permits a company to encode a program P as a new program $\widetilde{P}$ with the same behavior, but which does not reveal any of the inner workings of P. Obfuscation with certified deletion augments this with the ability to destructively measure $\widetilde{P}$, producing a certificate. If the certificate is valid, then the residual information from $\widetilde{P}$ should not be useful for evaluating the program further.

Theorem 1.1.1 (Informal)**.** *Assuming the existence of post-quantum indistinguishability obfuscation and post-quantum one-way functions, then there exists obfuscation with certified deletions for all (classical) programs.*

As part of the techniques developed for obfuscation with certified deletion, we obtain the first construction of primitives, e.g. encryption, with certified deletion where the verification key used to check the certificates can be safely made *public*. We also apply obfuscation with certified deletion to obtain new primitives with certified deletion. These include a public-key encryption scheme where the *secret decryption key* can be certifiably deleted, as well as a stronger version of a software copy-protection notion.

Theorem 1.1.2 (Informal)**.** *Assuming the existence of post-quantum indistinguishability obfuscation and post-quantum one-way functions, a variety of primitives, including obfuscation, functional encryption, and key revocation, exist with publicly verifiable certified deletion.*

Additionally, there exists a strong notion of secure software leasing.

Secret Sharing with Certified Deletion. Classically, threshold secret sharing allows a user to split a secret s into n shares $s_1, \ldots, s_n$. Any k shares can be used to reconstruct s, but any set of $< k$ shares reveals no information about s. Secret sharing with certified deletion augments this with the ability to destructively measure shares, producing certificates. Although stealing k shares would immediately reveal the secret in the classical setting, s remains hidden even if more shares are stolen, as long as the number of stolen shares minus the number of certifiably deleted shares is less than k.

We present two powerful, but incomparable ways of formalizing this: *adaptive* certified deletion and *no-signaling* certified deletion. We show how to construct threshold secret sharing under the adaptive definition and a more general notion under the no-signaling definition.

Theorem 1.1.3 (Informal)**.** *There exists threshold secret sharing with adaptive certified deletion. Furthermore, there exists secret sharing with no-signaling certified deletion for general monotone access structures.*

Signatures/NIZKs with Certified Deniability. Classically, signatures allow a receiver to verify who sent a specific message. [MPY24] showed how to verifiably destroy a signature so that the receiver cannot output a valid signature (with respect to the original verification procedure). Certified deniability strengthens this requirement to ensure that the receiver cannot produce *any* evidence that they previously saw a signature, after they produce a valid deletion certificate.

Similarly, non-interactive zero knowledge arguments (NIZKs) with certified deniability allow a prover to prove the truth of a statement x to a verifier, but after deletion, the verifier cannot convince anyone that they saw a proof of x.

Theorem 1.1.4 (Informal)**.** *There exist NIZKs with certified deniability in the quantum random oracle model (QROM). Furthermore, if one-way functions exist, then there exist signatures with certified deniability in the QROM.*

In contrast to prior notions of certified deletion, certified deniability is an application-independent paradigm with more comprehensive security guarantees. Whereas prior definitions disable specific capabilities upon deletion, certified deniability requires that after an object is deleted, it should be as if the user never received it.

1.2 Techniques

We begin with a quick review of the first technique used to certifiably delete information - in the original case, ciphertexts [BI20a, BK23]. The core idea is to encode the data using computational basis qubits, then mix some additional random Hadamard basis qubits in, which will act as checks. The ciphertext looks like

$$H^\theta |x\rangle = \bigotimes_i H^{\theta_i} |x_i\rangle \, , \; \mathsf{Enc}\left(\theta, b \oplus \bigoplus_{i:\theta_i=0} x_i\right) \tag{1.1}$$

To delete the ciphertext, measure $H^\theta |x\rangle$ in the Hadamard basis and output the result as the certificate cert. To verify cert, one can check that each bit cert_i corresponding to a Hadamard basis position (i.e. $\theta_i = 1$) matches x_i. Intuitively, since θ is hidden, anyone wanting to produce a valid certificate must measure almost the whole state $H^\theta |x\rangle$ in the Hadamard basis. However, measuring a computational basis state in the Hadamard basis destroys any information it used to contain.

For each of the later constructions, we develop new techniques that build on this basic idea to acquire additional properties.

1.2.1 Part I: Obfuscation

Intuitively, after deleting an obfuscated program, the adversary should not be able to evaluate it on *new* inputs. After all, clearly we cannot prevent them from remembering evaluations they already did. But we have no way of "looking inside the adversary's head" to identify which inputs they have already evaluated on. How do we define "new" inputs?

To do so, we draw inspiration from the classical notion of *differing inputs obfuscation* (diO). diO considers two programs P_0 and P_1 which differ at, say, a single input x^*, i.e. $P_0(x^*) \neq P_1(x^*)$. Knowledge of x^* would be enough to distinguish an obfuscation $\widetilde{P}_0$ of P_0 from an obfuscation $\widetilde{P}_1$ of P_1, since they have different behavior on x^*. diO guarantees that if x^* is hard to find, then $\widetilde{P}_0$ is indistinguishable from $\widetilde{P}_1$.

Any hard-to-find input x^* can be considered a "new" input. So we can consider P_0's behavior on x^* to be deleted if $\widetilde{P}_0$ looks like $\widetilde{P}_1$, even given x^*. Thus we arrive at the following definition:

1. Initialize the adversary with $\widetilde{P}_b$.
2. Receive a certificate cert from the adversary.
3. If cert is valid, send the adversary the differing input x^*.
4. The adversary outputs a guess b' for b.

Obfuscation with certified deletion requires that the adversary must guess b essentially at random: $\Pr[b = b'] = 1/2 + \mathsf{negl}(\lambda)$. In the full result, we allow the adversary to gain unbounded computational power in step 3, which allows them to find x^* themselves.

Missing Link: Repeated Partial Access. The requirements for obfuscation are significantly different from encryption. Encryption is an "all-or-nothing" primitive. Either the user has the key, in which case it can decrypt the message, or else it knows nothing about the message. In contrast, obfuscation requires *repeated access to partial information*: the ability to evaluate the program.

Prior to our work, no such technique existed. A crucial requirement for partial access is that *only* the prescribed information be required. After all, if an adversary could learn arbitrary functions of the encoded data, then nothing is hidden at all. To ensure that the decoded information matches the original in the BB84 state-based approach of [BI20a, BK23], one would need to ensure that the computational basis states match the original encoding. In other words, decide for any vector v whether $v_i = x_i$ for each i where $\theta_i = 0$. Unfortunately, repeated access to such an oracle immediately leaks the computational basis positions of x [Lut10].

Technique: Certified Deletion for Coset States. We overcome this issue by showing how to use *coset states* for certified deletion. A coset state $|S_{x,z}\rangle$ is an even superposition over elements of a coset $S + x$ of a subspace S, with some additional phase information attached:

$$|S_{\mathbf{x},\mathbf{z}}\rangle \propto \sum_{\mathbf{s}\in S} (-1)^{\mathbf{s}\cdot\mathbf{z}} |\mathbf{s}+\mathbf{x}\rangle$$

Measuring $|S_{x,z}\rangle$ in the Hadamard basis yields a vector in $S^\perp + z$. So, we can follow the same strategy of encoding the data in the computational basis while being able to verify measurements in the Hadmard basis for deletion. To encode a bit b, sample a random coset state along with a vector $\mathbf{r}$ and output[1]

$$|S_{\mathbf{x},\mathbf{z}}\rangle\,,\ \mathbf{r},\ b \oplus (\mathbf{x}\cdot\mathbf{r})$$

Notably, coset states remain unlearnable even when obfuscated programs which decide membership in $S + \mathbf{x}$ and $S^\perp + \mathbf{z}$ are publicly provided [CLLZ21a]. We extend this result to show that coset states can be certifiably deleted even in the presence of some side information $\mathcal{Z}(S, S + \mathbf{x}, S^\perp + \mathbf{z})$ which only partially hides $S + \mathbf{x}$ and $S^\perp + \mathbf{z}$.

One might hope that $\mathcal{Z}$ could contain obfuscated membership programs for $S + \mathbf{x}$ and $S^\perp + \mathbf{z}$, similarly to [CLLZ21a]'s result. Although this does have some deletion properties, ultimately we wish to obtain security against an adversary who against unbounded computational power after deletion. If $\mathcal{Z}$ only computationally hid $S + \mathbf{x}$, then the adversary could directly extract S and $\mathbf{x}$ from it after deletion, revealing the encoded bit.

To solve this problem, we use instead allow $\mathcal{Z}$ to use a random super-coset $T + \widetilde{x} \supset S + x$. By setting T to be exponentially small than the overall space, but exponentially larger than S, we obtain two useful properties. First, since T is much smaller than $\mathbb{F}_2^\lambda$, it is hard to find a vector in $T + \widetilde{x} \backslash S + x$. So testing $v \in T + \widetilde{x}$ is essentially as good as testing $v \in S + x$. Second, since T is much larger than S, $T + \widetilde{x}$ hides enough information about x than $x \cdot r$ is uniformly random, even given $T + \widetilde{x}$ and r. Then we are able to show that if $\mathcal{Z}(S, T + \widetilde{x}, S^\perp + \mathbf{z})$ is computationally simulateable using only $(S, T + \widetilde{x}, R^\perp + \widetilde{\mathbf{z}})$ for a random super-coset $R^\perp + \widetilde{\mathbf{z}} \supset S^\perp + \widetilde{\mathbf{z}}$, then the encoded bit is information-theoretically deleted.

Constructing Obfuscation With Certified Deletion. Armed with a technique for repeated partial access followed by deletion, we construct obfuscation with certified deletion. First, encrypt (with certified

[1] Technically, $\mathbf{x}$ should be a *canonical representative* for the coset $S + \mathbf{x}$ so that the encoding uniquely specifies b. See Part I for full details.

deletion) the program P using the coset state scheme. This ciphertext can be decrypted using a computational basis measurement of the ciphertext. Then, obfuscate a classical program that has the secret key hardcoded. The program takes as input a measurement result v and an input a. It checks that v is likely to be valid measurement result of the ciphertext, i.e. that $v \in T + \widetilde{x}$. As mentioned previously, this is almost as good as checking that $v \in S + \mathbf{x}$. If so, it obtains P by decrypting v with the hard-coded secret key, then evaluates and outputs $P(a)$.

The obfuscated classical program can be considered as the side-information $\mathcal{Z}$, so we can prove security using the previously discussed technique for deleting coset states. For a more detailed overview, see Chapter 4.

1.2.2 Part II: Secret Sharing

Secret sharing relies on the assumption that no adversary can steal k shares, which would be enough to reconstruct the secret. Intuitively, if an adversary steals a deleted share - or pretends to delete one it has already stolen - then it should still need to steal an additional k shares to recover the secret.

We formalize this intuition by considering an adversary which steals shares one at a time, subject to the constraint that they never control k shares which have not yet been deleted. This adversary should not be able to distinguish a secret sharing of s from a secret sharing of s', even if they eventually steal all of the shares.[2]

Challenge: Preventing Gradual Accumulation of Information. The primary challenge for deleting secret shares is to prevent the gradual accumulation of information as the adversary acquires more shares. In the adversarial model, the adversary may continually steal new shares after producing valid certificates for shares it already stole. If part of each share were not deletable, e.g. because it was classical, then those parts would accumulate over time. Eventually, they might allow the adversary to violate security by recovering the secret.

Indeed, in Part II we show a natural construction based on the BB84-based approach of [BK23] which has some deletion properties, but still completely reveals the secret after enough shares are stolen and deleted. The core issue with the BB84-based construction is that knowledge of the basis θ is necessary to decode the message. This classical information must be somehow hidden in a way that can be deleted, or else the adversary can eventually steal it. Unfortunately, hiding θ so that it can be deleted share-by-share seems to already require secret sharing with certified deletion.

Technique: Basis-Independent Decoding. We observe that the use of error-correcting codes can help in reconstructing from the BB84-based encoding scheme without using the basis θ. Concretely, consider encoding a message m as an error-correcting codeword $c = [c_1, \ldots, c_t]$. We can write the individual symbols as computational basis states $|c_i\rangle$. Next, we replace some of these computational basis states with random Fourier basis states and keep the description of the replacements as the verification key:

$$\begin{array}{rccccccccccc} \text{Encoding} = & |c_1\rangle & \otimes & |c_2\rangle & \otimes & \mathsf{QFT}\,|r_1\rangle & \otimes & |c_4\rangle & \otimes & \mathsf{QFT}\,|r_2\rangle & \otimes & \ldots \\ \mathsf{vk} = & * & & * & & r_1 & & * & & r_2 & & \ldots \end{array} \tag{1.2}$$

A computational basis measurement of this state consists of many parts c_i of the codeword mixed with a few random errors in the replaced positions. If the number of replacements is within the error-correction capability of the code, the message m can be recovered. On the other hand, individually measuring

[2] We call this adaptive certified deletion. We also define an incomparable, "no-signaling" definition in Part II and construct a scheme satisfying it. This overview focuses on the more technically involved construction for adaptive security.

each position in the Fourier basis destroys the codeword if it was a computational basis position, or can be checked by comparing it to the verification key if it was a Fourier basis position. For certain error correcting codes, if the encoding is divided up into appropriate chunks which are individually deleted (since the whole state together reveals m), we can hope that this destroys m.

Although we prove certified deletion only for our specific construction, we believe this technique is also secure for a more general setting. Informally, we use three properties to show security: (1) any set of symbols below the reconstruction threshold look uniformly random, (2) individual symbols c_i can be reverse-sampled from the other symbols, and (3) the reverse-sampling procedure forms a "good" randomness extractor (see the next paragraph).

Technique: High-Rate Seedless Randomness Extraction. As part of proving certified deletion in the polynomial error-correcting code, we need to introduce a new way to extract large amounts of entropy from deleted shares. Previous works also used a seedless extractor in their analysis, but use a large entropy source to extract just a single bit of randomness. For secret sharing, we need a more efficient solution.

Roughly, in the analysis we will generate a minimal set of shares, then reverse sample the rest to ensure consistency. After deleting a share, that share has high entropy in the adversary's view. If the reverse sampling procedure makes good use of that entropy, then the next share that the adversary steals will look uniformly random. Since the adversary may delete and steal shares in a 1-to-1 manner, the extractor/reverse sampler needs to produce (almost) a full share of randomness using the entropy from a single deleted share.

We show that polynomial interpolation over finite fields is a good randomness extractor for deleted shares, and that it produces almost the same amount of entropy as the input. Polynomial interpolation is precisely the reverse sampling procedure for Reed-Solomon codes.

As a further optimization, this result applies to extension fields, e.g. $\mathbb{F}_{2^n}$. The quantum Fourier transform over $\mathbb{F}_{2^n}$ is simply applying a Hadamard gate to each qubit of the element. This enables us to remove entanglement in our construction by encoding individual qubits in either the computational basis or the Hadamard basis.

Constructing Secret Sharing with Certified Deletion. Our construction uses the observation that Shamir's polynomial secret sharing also has good error-correcting properties. We instantiate the outline above by sampling a random polynomial f such that $f(0) = s$ is the secret. Then shares consist of some number of evaluations of f mixed with a smaller number of Fourier basis states. By carefully tuning the parameters, we ensure that any k shares are together enough to correct the errors induced by the Fourier positions, but any $k - 1$ shares look uniformly random.

See Chapter 11 for a more detailed overview.

1.2.3 Part III: Signatures and NIZKs with Certified Deniability

Certified deniability is a new, application-independent paradigm for deletion. Intuitively, it realizes the comprehensive notion that

> *Once the adversary deletes the cryptographic object, it looks as if they never received it in the first place.*

We formalize this for signatures by requiring that there is a simulator which does not receive *any* signatures, but which can simulate the final view of an adversary who does receive a signature then deletes it.

In the quantum random oracle model (QROM), the simulator is further restricted to not be able to choose the random oracle, similar to the classical notion of deniability [Pas03].[3]

Deniability versus Deletion. In certified deniability, we propose a fundamentally new paradigm from prior works on certified deletion. Previously, certified deletion was defined by identifying specific abilities the adversary obtains from the encoded data – for example, the ability to evaluate an obfuscated program – then specifically disallowing that ability after the encoded information (e.g. the program) is certifiably deleted. Unfortunately, tailoring certified deletion to each application in this way can miss strategies that are not explicitly disallowed, but still give an undesirable result. For example [MPY24]'s construction of revocable signatures disallows an adversary who receives a signature from outputting both a valid certificate and a valid signature. One might hope that this prevents the adversary Bob from proving that the signer Alice signed something after deleting her signature. However, we show in Part III that Bob could still provide irrefutable evidence of what Alice signed, even after producing a valid certificate.

In contrast, certified deniability ensures that anything that could be learned using a deleted signature could be learned without it. This immediately rules out evidence collection attacks due to signature unforgeability, as well as other attacks that are not explicitly considered by the definition.

We emphasize that the simulator-based definition does not specifically use any properties of signatures. In principle, certified deniability could be applied to *any* cryptographic primitive with little modification.

Challenge: Non-Programmability. The main barrier to achieving certified deniability in the QROM is the restriction that the simulator cannot choose the random oracle. Traditionally, achieving simulation for one-round protocols, such as in non-interactive zero-knowledge arguments, requires choosing the random oracle's behavior at certain inputs. Classically, any simulator which internally used a different random oracle H' would produce a view that does not line up with the real random oracle H, rendering it immediately distinguishable from a real adversary's view. In fact, Pass [Pas03] uses this observation to show that non-interactive zero knowledge is *impossible* classically, even in the random oracle model. A similar proof works for deniable signatures in the classical setting.

Technique: Forgetful Local Reprogramming. We use uniquely quantum phenomena to bypass Pass's impossibility. Intuitively, we will use ideas previously developed for certified deletion to force the adversary to "forget" the points x where it evaluated the random oracle on. For example, we could prepare a subspace state and coherently evaluate the random oracle H to obtain

$$\sum_{x\in S} |x\rangle \otimes |H(x)\rangle$$

If the adversary returns the whole state, it must have forgotten every $x \in S$. Therefore it must *also* forget the behavior of the random oracle on x. So even if the simulator internally uses a false random oracle H' with $H'(x) \neq H(x)$ (over $x \in S$) to run the adversary on $\sum_{x\in S} |x\rangle \otimes |H'(x)\rangle$, the adversary's final output looks as though it actually accessed the correct oracle H, since they are forced to forget how the oracle behaved on $x \in S$.

Constructing Signatures and NIZKs. To construct signatures with certified deniability, we sign the message in superposition using the randomness $H(x)$ under a carefully chosen signature scheme:

$$\sum_{x\in S} |x\rangle \otimes |\mathsf{Sign}_k(m; H(x))\rangle$$

[3] See Chapter 19 for a more detailed justification of this restriction.

Fiat-Shamir signatures have the useful property that they can be simulated, without a real signature, if one controls the random oracle H. So the simulator can apply the forgetful local reprogramming technique to locally simulate a signature $\sum_{x\in S}|x\rangle\otimes|\mathsf{Sign}_k(m;H'(x))\rangle$ using a chosen H', while ensuring that its usage of H' instead of H goes undetected in the final view.

NIZKs with certified deniability can also be constructed using essentially the same idea. See Chapter 19 for a more detailed overview.

Chapter 2

Preliminaries

We begin with a few preliminary definitions that will be useful across all of the works presented. Individual parts will also have their own preliminary sections that cover topics directly relevant to them. Unless otherwise noted, all hardness assumptions are against quantum adversaries.

2.1 General Notation

$\mathbb{F}$ denotes a generic field. $\mathbb{F}_2 = \{0, 1\}$ denotes the binary field. Vectors $\mathbf{x}$ are denoted by bold font. $\mathbf{e}_i$ denotes the i'th standard basis vector which is 1 at index i and 0 elsewhere.

2.2 Cryptographic Notation

The security parameter is λ. Occasionally we will use n to represent a general parameter; it may be assigned as λ, but this is not necessary. A function $\mu(\lambda) : \mathbb{N} \to [0, 1]$ is **negligible** if $\mu(\lambda) = o(1/p(\lambda))$ for all polynomials p. $\mathsf{negl}(\lambda)$ denotes a generic negligible function. Two distributions D_0 and D_1 are **computationally indistinguishable**, denoted as $D_0 \approx_c D_1$, if for all quantum polynomial time (QPT) algorithms Adv, Adv cannot distinguish between samples from D_0 and D_1 with better than $\mathsf{negl}(\lambda)$ advantage:

$$|\Pr[\mathsf{Adv}(d_0) : d_0 \leftarrow D_0] - \Pr[\mathsf{Adv}(d_1) : d_1 \leftarrow D_1]| = \mathsf{negl}(\lambda)$$

D_0 and D_1 are **statistically indistinguishable**, denoted as $D_0 \approx_s D_1$ if this holds against algorithms running in arbitrary time and space (referred to as "unbounded" algorithms). When we wish for a finer-grained treatment, we say that D_0 and D_1 are computationally ϵ**-indistinguishable**, denoted by $D_0 \approx_c^\epsilon D_1$, if the advantage is instead bounded by ϵ.

A distribution $D(\cdot)$ is **semantically hiding** if for every m_0 and m_1, $D(m_0) \approx_c D(m_1)$.

2.3 Quantum Computation

General Quantum Computation. A **register** $\mathcal{X}$ is associated with a Hilbert space $\mathscr{H}$ and contains quantum states. An n-qubit quantum state exists in the Hilbert space $\mathscr{H} = (\mathbb{C}^2)^{\otimes n}$. A **pure state** $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$ can be represented as a linear combination of the standard basis vectors $\{|x\rangle : x \in \{0, 1\}^n\}$ with unit length. A **mixed state** ρ represents a probability distribution over pure states; it can be represented as the density matrix $\sum_i |\psi_i\rangle \langle\psi_i|$. Here, $\langle\psi| = |\psi\rangle^\dagger$. More generally, we will also consider q*d*its, which

are quantum states over a larger alphabet. For any set S, an S-qudit lies in the Hilbert space $\mathbb{C}^{|S|}$ with standard basis $\{|x\rangle : x \in S\}$. $|x\rangle^{\mathcal{X}}$ denotes a quantum state $|x\rangle$ stored in register $\mathcal{X}$.

A **quantum operation** F is a completely-positive trace-preserving (CPTP) map from a register $\mathcal{X}$ to a register $\mathcal{Y}$. We denote that an operation U is applied to register $\mathcal{X}$ by $U^{\mathcal{X}}$. When applying the operation $U^{\mathcal{X}} \otimes I^{\mathcal{Y}}$, which acts as the identity on register $\mathcal{Y}$, we often omit the identity operator and just write $U^{\mathcal{X}}$. The most common type of operation is a **unitary** operation, which is an operation $U : \mathcal{X} \to \mathcal{X}$ such that $UU^\dagger = I$ is the identity operation. Unitaries can therefore be inverted by applying $U^\dagger$.

Another common operation is a measurement. A projector $\Pi : \mathcal{X} \to \mathcal{X}$ is a quantum operation such that $\Pi\Pi = \Pi$. A **projective measurement** (PVM) is a set of mutually orthogonal projectors $\{\Pi_x : x \in S\}$ such that $\sum_{x \in S} \Pi_x = I$. S is the set of measurement outcomes and Π_x is associated with result x. The probability of obtaining outcome $x \in S$ when measuring a state ρ is $\mathsf{Tr}[\Pi_x \rho \Pi_x]$.

The **fidelity** of two pure states $|\psi\rangle$ and $|\phi\rangle$ is $|\langle\psi|\phi\rangle|^2$. It represents the similarity between $|\psi\rangle$ and $|\phi\rangle$. **Trace distance** is a notion of distance between two quantum states ρ and σ.

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

The trace distance upper bounds the probability of any (unbounded) algorithm distinguishing between ρ and σ. In the case of pure states $|\psi\rangle$ and $|\phi\rangle$, the trace distance can be represented in terms of the fidelity:

$$\mathsf{TD}[|\psi\rangle, |\phi\rangle] = \sqrt{1 - |\langle\psi|\phi\rangle|^2}$$

Quantum states may also be represented in other bases. Common bases for a single qubit are the **computational basis** $\{|0\rangle, |1\rangle\}$ and the **Hadamard basis** $\{|+\rangle, |-\rangle\}$, where

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

The **Hadamard** transformation H maps between the computational and Hadamard basis: $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$.

The **Pauli operators** are a set of 3 unitaries X, Y, and Z, where X maps $|b\rangle \mapsto |1 - b\rangle$ for $b \in \{0, 1\}$, Z maps $H|b\rangle \mapsto H|1 - b\rangle$ for $b \in \{0, 1\}$, and $Y = iXZ$.

A **non-uniform quantum polynomial-time** (QPT) algorithm consists of an unitary U which can be implemented in polynomial time and a register $\mathcal{A}$ containing a non-uniform quantum state. On input a quantum state contained in register $\mathcal{X}$, it outputs $U(\mathcal{X} \otimes \mathcal{A})$. In the case where the input contained in $\mathcal{X}$ depends in some way on the auxiliary state contained in $\mathcal{A}$, the contents of $\mathcal{X}$ and $\mathcal{A}$ may be entangled.

Lemma 2.3.1 (Gentle Measurement Lemma [Win99, CMSZ22])**.** *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 $\mathsf{TD}(\rho, \rho') \leq 2\sqrt{\delta}$.

Subspace States. For any subspace $A \subset \mathbb{F}_2^n$, the **subspace state** $|A\rangle$ for A is

$$|A\rangle :\propto \sum_{a \in A} |a\rangle$$

A useful fact about subspace states is that applying a Hadamard transform to $|A\rangle$ results in a subspace state over the dual $A^\perp = \{x \in \mathbb{F}_2^n : x \cdot a = 0 \; \forall a \in A\}$:

$$H^{\otimes n} |A\rangle = |A^\perp\rangle \propto \sum_{x \in A^\perp} |x\rangle$$

Aaronson and Christiano [AC12a] show that the projector onto $|A\rangle$ can be implemented using queries to membership oracles O_A and $O_{A^\perp}$ that decide the query's membership in A (respectively, $A^\perp$).

Lemma 2.3.2 ([AC12a])**.** *Let $A \subset \mathbb{F}_2^n$ be a subspace. Let $\Pi_A = \sum_{a \in A} |a\rangle \langle a|$ and $\Pi_{A^\perp} = \sum_{a \in A^\perp} |a\rangle \langle a|$ be projectors onto the space spanned by elements of A and $A^\perp$, respectively. Then*

$$H^{\otimes n} \Pi_{A^\perp} H^{\otimes n} \Pi_A = |A\rangle \langle A|$$

Ben-David and Sattath show that given a random subspace state of appropriate dimension, it is hard to find a vector $v_1 \in A \backslash \{0\}$ together with a vector $v_2 \in A^\perp \backslash \{0\}$, even when given oracle access O_A and $O_{A^\perp}$.

Lemma 2.3.3 ([BS23])**.** *Let $A \subset \{0,1\}^\lambda$ be a random subspace of dimension $\lambda/2$ and let $\epsilon > 0$ be such that $1/\epsilon = o(2^{\lambda/2})$. Given one copy of $|A\rangle$ and oracle access to O_A and $O_{A^\perp}$, any adversary who produces $v_1 \in A \backslash \{0\}$ together with $v_2 \in A^\perp \backslash \{0\}$ with probability ϵ requires $\Omega(\sqrt{\epsilon} 2^{\lambda/2})$ queries.*

It is also useful to consider **coset states**, which are formed by applying Pauli X and Z operations to subspace states:

$$|S_{x,z}\rangle \coloneqq X^x Z^z |S\rangle \propto \sum_{s \in S} (-1)^{s \cdot z} |s + x\rangle$$

They can also be thought of as superpositions over a coset $S + x$ in the computational basis and $S^\perp + z$ in the Hadamard basis. In fact, applying the Hadmard transform to a coset state gives the dual coset state:

$$H^{\otimes n} |S_{x,z}\rangle = |S^\perp_{z,x}\rangle$$

Part I

Obfuscation with Certified Deletion

In this part, we show how to verifiably delete programs. As mentioned previously, the ability to delete programs could be very useful for preventing piracy in program rentals. Ideally, we would like to design a protocol where a developer Alice could lease her software to a client Bob by shipping a copy to him. Bob can evaluate the software freely on any input he wants, while Alice charges him monthly 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.

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.

Chapter 3

Results

As our main result, we define and construct obfuscation with certified deletion. We take inspiration from the classical notion of differing inputs obfuscation (diO) [BGI+01] to define *differing inputs obfuscation with certified deletion* (diO-CD). Loosely speaking, diO-CD satisfies the standard notion of differing inputs obfuscation, 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).

Theorem 3.0.1 ((Informal))**.** *Assuming indistinguishability obfuscation and injective one-way functions, there exists* diO-CD *for all differing inputs circuits families with a polynomial number of differing inputs.*

As our main technical tool, we develop a technique for deleting *coset states* [VZ21, CLLZ21a]. Crucially, our technique allows for *repeated access* to partial information about the data, followed by information-theoretic deletion of whatever is left.

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 sub-exponential diO-CD and one-way functions. The latter follows from combining diO-CD with post-quantum public-key encryption and injective one-way functions. Functional encryption with key revocation is the *only* result from this section 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.

Chapter 4

Technical Overview

4.1 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

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

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 [AC12b] 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 $\mathbf{x}, \mathbf{z} \in \mathbb{F}_2^n$, and is written as

$$|S_{\mathbf{x},\mathbf{z}}\rangle \coloneqq \frac{1}{\sqrt{|S|}} \sum_{\mathbf{x}' \in S+\mathbf{x}} (-1)^{\mathbf{z}\cdot\mathbf{x}'} |\mathbf{x}'\rangle .$$

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

$$|S_{\mathbf{x},\mathbf{z}}\rangle\,,\ \mathsf{Enc}(S,\ \mathbf{r},\ b \oplus (\mathbf{x}\cdot\mathbf{r})),$$

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

Additionally, it is possible to check whether $\mathbf{x}'$ has been tampered with by verifying that $\mathbf{x}' \in S + \mathbf{x}$. It is even possible to publish an oracle for this consistency check, *without* leaking S and $\mathbf{x}$ [CLLZ21a]. 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 $\mathbf{x}$ hard-coded. Unfortunately, the obfuscation only hides S and $\mathbf{x}$ computationally. After deletion, an unbounded adversary could learn $\mathbf{x}$, $\mathbf{r}$, and $\widetilde{b} \oplus (\mathbf{x}\cdot\mathbf{r})$, 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 $T + \widetilde{\mathbf{x}}$ that contains $S + \mathbf{x}$. We set $\dim(T) = 3\lambda/4$ as a happy medium,

which has two nice properties. First, since T is a negligible fraction of $\mathbb{F}_2^\lambda$, it is hard for an adversary to find a vector in $T + \widetilde{\mathbf{x}} \setminus S + \mathbf{x}$, so the consistency check will be essentially as good as using $S + \mathbf{x}$. Second, since S is a negligible fraction of T, $T + \widetilde{\mathbf{x}}$ statistically hides enough information about $\mathbf{x}$ that $\mathbf{x} \cdot \mathbf{r}$ is uniformly random, even given $T + \widetilde{\mathbf{x}}$. Therefore we can hope to get information-theoretic deletion.

4.2 Coset Framework

Before describing our main certified deletion theorem, we discuss some tools which we develop to help analyze coset states. The first tool is a new canonical representative $\mathsf{Can}_S(\mathbf{x})$ for cosets $S + \mathbf{x}$ which has more algebraic structure. Previously, [CLLZ21b] defined canonical representatives lexicographically, which lacks any sort of algebraic structure. The algebraic structure of our coset representatives allows a more flexible and sophisticated approach.

As a second tool, we also show a method to delay the preparation of a coset state $|S_{\mathbf{x},\mathbf{z}}\rangle$ using EPR pairs, similarly to a technique used to analyze BB84 states [BB84].

Complementary Representatives. We define the canonical representative $\mathsf{Can}_S(\mathbf{x})$ of a coset $S + \mathbf{x}$ using a complementary subspace $C \subset \mathbb{F}_2^n$ to S. In other words, $C \cap S = \{0\}$ and $\mathsf{Span}(S, C) = \mathbb{F}_2^n$. A useful fact about complementary subspaces is that any vector $\mathbf{x} \in \mathbb{F}_2^n$ can be uniquely decomposed as the sum of an element of S and an element of C, say $\mathbf{x} = \mathbf{x}_S + \mathbf{x}_C$. We define

$$\mathsf{Can}_{S,C}(\mathbf{x}) \coloneqq \mathbf{x}_C$$

There are many possible choices of C for each S. Making matters more complicated, $S^\perp$ may not be complementary to S since the vector space is over a finite field. To simplify matters, we consider the description of S to also specify C, enabling the notation $\mathsf{Can}_S(\mathbf{x})$.

Properties of the Representatives. An initial observation is that the set of canonical representatives $\{\mathsf{Can}_S(\mathbf{x}) : \mathbf{x} \in \mathbb{F}_2^n\}$ is precisely C. Due to this, $\mathsf{Can}_S(\cdot)$ is homomorphic, i.e. $\mathsf{Can}_S(\mathbf{x}) + \mathsf{Can}_S(\mathbf{z}) = \mathsf{Can}_S(\mathbf{x} + \mathbf{z})$. The space C is also isomorphic to the quotient group $\mathbb{F}_2^n / S = \{S + \mathbf{x} : \mathbf{x} \in \mathbb{F}_2^n\}$.

There are two more sophisticated properties that will also be useful in our analysis of certified deletion. These properties deal with how $\mathsf{Can}_S(\cdot)$ behaves when altering the space that S lies within.

- **Embedding.** If we increase the size of the space by adjoining some C' to C — for example to move from $\mathbb{F}_2^n$ to $\mathbb{F}_2^{2n}$ — then the canonical representative of $S + \mathbf{x}$ with respect to $\mathsf{Span}(C, C')$ does not change from $\mathsf{Can}_S(\mathbf{x})$. In fact, for any $\mathbf{c}' \in C'$, the canonical representative of $S + (\mathbf{x} + \mathbf{c}')$ in the expanded space is simply

$$\mathsf{Can}_{S,\mathsf{Span}(C,C')}(\mathbf{x} + \mathbf{c}') = \mathsf{Can}_{S,C}(\mathbf{x}) + \mathbf{c}'$$

- **Rerandomization.** $\mathsf{Can}_S(\cdot)$ behaves particularly simply under rerandomization of the space $\mathbb{F}_2^n$ by a change-of-basis M. M maps any coset $S + \mathbf{x}$ to $MS + M\mathbf{x}$ and maps C to MC. If we are given the canonical representative before the rerandomization, we can compute the canonical representative of the rerandomized coset as

$$\mathsf{Can}_{MS}(M\mathbf{x}) = M\mathsf{Can}_S(\mathbf{x})$$

Delayed Preparation of Coset States. Historically, the ability to delay the preparation of BB84 states has been quite useful. It allows a sender to send some BB84 states to a receiver, then decide the basis *later*. Accordingly, the receiver's behavior must be independent of the chosen basis.

We show how to perform a similar trick for coset states. First, the sender prepares some number of EPR pairs $\sum_{\mathbf{v}\in\mathbb{F}_2^n} |v\rangle_{\mathcal{S}} \otimes |\mathbf{v}\rangle_{\mathcal{R}}$ and sends $\mathcal{R}$ to the receiver. It keeps the sender register $\mathcal{S}$. Later, it can choose a subspace $S \subset \mathbb{F}_2^n$ and measure $\mathcal{S}$ to obtain $\mathbf{x}$ and $\mathbf{z}$, which collapses the receiver's state to the coset state $|S_{\mathbf{x},\mathbf{z}}\rangle$. Specifically, it computes $\mathsf{Can}_S(\cdot)$ on register $\mathcal{S}$ in the computational basis and measures the result, then computes $\mathsf{Can}_{S^\perp}(\cdot)$ on register $\mathcal{S}$ in the Hadamard basis and measures the result (the order of these does not matter). This technique actually works for any definition of canonical coset representatives.

4.3 Certified Deletion for Coset States

Next, we describe the general compiler for deleting coset states. First, consider a simple template for certified deletion: to hide a bit b, we give the adversary the following state:

$$|S_{\mathbf{x},\mathbf{r}}\rangle \quad \text{and} \quad \mathcal{Z}(S,\ \mathbf{r},\ b \oplus (\mathsf{Can}_S(\mathbf{x})\cdot\mathbf{r})),$$

where $|S_{\mathbf{x},\mathbf{z}}\rangle$ 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.

To prove deletion, measure the subspace coset state in the Hadamard basis to get a vector $\mathbf{z}' \in S^\perp + \mathbf{z}$. If done honestly, this destroys essentially all information about $\mathbf{x}$ and removes b from their view. We will hope to prove that *any* strategy an (efficient) adversary uses to obtain a $\mathbf{z}' \in S^\perp + \mathbf{z}$ will also statistically remove b from their view.

[BK23] showed how to prove this when $\mathcal{Z}$ satisfies *semantic security* with respect to S and S is the span of standard basis vectors. However, as discussed previously, we want to use more information about $\mathbf{x}$ in our primitives. For example, to do the noisy check, $\mathcal{Z}$ might output a random super-coset $T + \widetilde{x} \supset S + \mathbf{x}$. Unfortunately, revealing T means $\mathcal{Z}$ cannot semantically hide S,[1] so [BK23]'s proof falls short.

Conditions for $\mathcal{Z}$. We show more flexible conditions for $\mathcal{Z}$ under which we can obtain certified deletion. Specifically, consider $\mathcal{Z}(S, T + \widetilde{\mathbf{x}}, S^\perp + \mathbf{z}, \mathbf{r}, \widetilde{b})$, which can depend on information about $\mathbf{x}$ and $\mathbf{z}$. The notion of hiding we need from $\mathcal{Z}$ is inspired by [CLLZ21b]'s application of subspace-hiding [Zha19b] to cosets. Roughly, a distribution $\mathcal{D}$ is *coset-hiding* if $\mathcal{D}(S + \mathbf{x})$ cannot be distinguished from a simulated distribution $\mathsf{Sim}(W + \mathbf{x}')$ which depends only on a random super-coset $W + \mathbf{x}' \supset S + \mathbf{x}$.

With coset-hiding defined, we can state the result. We show that if $\mathcal{Z}$ semantically hides its first input S and coset-hides its third input $S^\perp + \mathbf{z}$ – that is, it can be simulated using only $(T + \widetilde{x}, R^\perp + \widetilde{vz}, \mathbf{r}, \widetilde{b})$ – then the following experiment information-theoretically hides the encoded bit b:

1. Initialize the adversary with

   $$|S_{\mathbf{x},\mathbf{r}}\rangle \quad \text{and} \quad \mathcal{Z}(S, T + \widetilde{\mathbf{x}}, S^\perp + \mathbf{z}, \mathbf{r}, b \oplus (\mathsf{Can}_S(\mathbf{x})\cdot\mathbf{r}))$$

   where $T + \widetilde{\mathbf{x}}$ is a random super-coset of $S + \mathbf{x}$.

2. Receive a certificate cert from the adversary. If cert $\in S^\perp + \mathbf{z}$, then output the adversary's view. Otherwise, output $\perp$.

[1] At least, not obviously. See the discussion later in this section.

Since the experiment outputs $\bot$ whenever the adversary outputs an invalid certificate, the output of the experiment only carries whatever information they can keep while also outputting a valid certificate.

Proof Overview: Dealing with $T + \widetilde{\mathbf{x}}$. We divide the proof into two parts: one where the primal super-coset $T + \widetilde{\mathbf{x}}$ appears in $\mathcal{Z}$, and another where the dual coset $S^\perp + \mathbf{z}$ appears.[2] We begin by discussing the approach to dealing with $T + \widetilde{\mathbf{x}}$, i.e. a certified deletion theorem for $\mathcal{Z}(S, T + \widetilde{\mathbf{x}}, \mathbf{r}, \widetilde{b})$.

We will reduce to a certified deletion theorem for cosets where the side information $\mathcal{Z}'$ depends only on $(S, \mathbf{r}, \widetilde{b})$, but *not* on $T + \widetilde{\mathbf{x}}$. This is similar to [BK23]'s result, but we can also begin with the the theorem for dual-coset leakage to get a final result that incorporates both $T + \widetilde{\mathbf{x}}$ and $S^\perp + \mathbf{z}$.

The apparent difficulty is how to incorporate $T + \widetilde{\mathbf{x}}$ into $\mathcal{Z}$ without receiving any information about T or $\widetilde{x}$ from $\mathcal{Z}'(S, \mathbf{r}, \widetilde{b})$. To solve this, we actually start with the original certified deletion theorem over a smaller space $\mathbb{F}_2^{3\lambda/4}$, then embed the whole space inside $\mathbb{F}_2^{\lambda}$. Next, we rerandomize the whole space under a change-of-basis M, which maps $\mathbb{F}_2^{3\lambda/4}$ to a random subspace T that contains the re-randomized space MS. At this point, we can sample a random $\widetilde{\mathbf{x}}$ and be assured that $T + \widetilde{x} \supset S + (M\mathbf{x} + \widetilde{\mathbf{x}})$ since $S + M\mathbf{x} \subset T$.

The rest of the challenge $|S_{\mathbf{x},\mathbf{z}}\rangle$ and $\mathcal{Z}'(S, \mathbf{r}, \widetilde{b})$ can now be doctored to match the rerandomized cosets $T + \widetilde{x}$ and $S + (M\mathbf{x} + \widetilde{\mathbf{x}})$. First, we can use M and $\widetilde{x}$ to modify $|S_{\mathbf{x},\mathbf{z}}\rangle$ into $|(MS)_{M\mathbf{x}+\widetilde{\mathbf{x}},\mathbf{z}'}\rangle$ matching $S + (M\mathbf{x} + \widetilde{\mathbf{x}})$. Then, by using the embedding and rerandomization properties of $\mathsf{Can}_S(\cdot)$, we can doctor $\mathbf{r}$ and $\widetilde{b}$ so that $(|(MS)_{M\mathbf{x}+\widetilde{\mathbf{x}},\mathbf{z}'}\rangle, MS, T + \widetilde{\mathbf{x}}, \mathbf{r}', \widetilde{b}')$ encodes the same bit as before. This is enough to construct $\mathcal{Z}(MS, T + \widetilde{\mathbf{x}}, \mathbf{r}', \widetilde{b}'))$, so any adversary breaking the expanded encoding (which matches the statement we want to prove) also breaks the original certified deletion theorem.

Proof Overview: Dealing with $S^\perp + \mathbf{z}$. Next, we discuss how to deal with $S^\perp + \mathbf{z}$ appearing the side-information (without $T + \widetilde{x}$. We first claim that the side-information $\mathcal{Z}(S, S^\perp + \mathbf{z}, \mathbf{r}, b \oplus (\mathsf{Can}_S(\mathbf{x}) \cdot \mathbf{r})$ statistically hides b. This includes no information about $\mathbf{x}$, and b is masked by $(\mathsf{Can}_S(\mathbf{x}) \cdot \mathbf{r}$. Due to the homomorphism of $\mathsf{Can}_S(\cdot)$, this is a random bit with overwhelming probability over $\mathbf{r}$.

Of course, the adversary's view also includes $|S_{\mathbf{x},\mathbf{z}}\rangle$. Together the coset state and $\mathcal{Z}$ computationally determine b. We will show that any adversary producing $\mathsf{cert} \in S^\perp + \mathbf{z}$ essentially must have completely measured $|S_{\mathbf{x},\mathbf{z}}\rangle$ in the Hadamard basis, destroying all information about $\mathbf{x}$ and thereby b.[3]

To show this, instead of directly sampling S, $\mathbf{x}$, and $\mathbf{z}$ and giving the adversary $|S_{\mathbf{x},\mathbf{z}}\rangle$, we delay its preparation using the approach described previously. This results in the challenger and adversary sharing EPR pairs $\sum_{\mathbf{v}} |\mathbf{v}\rangle^{\mathcal{C}} \otimes |\mathbf{v}\rangle^{\mathcal{R}}$. If the adversary were truly to collapse their half to $H^{\otimes\lambda}|\mathsf{cert}\rangle$, then the challenger's half would collapse to the same. This is efficiently testable using register $\mathcal{C}$, so the probability of it occurring is the same even if we utilize the coset-hiding properties of $\mathcal{Z}$ to switch $S^\perp + \mathbf{z}$ for $R^\perp + \widetilde{\mathbf{z}}$. Finally, we observe that the certificate check $\mathsf{cert} \in S^\perp + \mathbf{z}$ can be replaced by checking that $\mathcal{C}$ is supported on basis vectors $\mathbf{z}'$ where $(\mathsf{cert} - \mathbf{z}') \in S^\perp$. We can show that if $\mathbf{z}'$ where *not* cert, then the probability of this check passing is negligible, by considering the primal spaces S and $R = (R^\perp)^\perp$. S is sampled *after* cert by adjoining $\lambda/4$ random vectors $\mathbf{s}_i$ to R. If $(\mathsf{cert} - \mathbf{z}') \in S^\perp$, then $(\mathsf{cert} - \mathbf{z}') \cdot \mathbf{s}_i = 0$. However, this occurs independently for each $\mathbf{s}_i$ with $1/2$ probability, so the probability of it holding for all of them is negligible.

[2] It is possible to do these together, but dividing them brings some additional insights.

[3] This is a much stronger property than shown in [BK23], which only guarantees that *most* of the qubits were measured in the Hadamard basis.

4.4 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, including an algorithm for *delayed preparation of subspace coset states*. Chapter 6 presents these results.

Our first contribution is to define a canonical representative $\mathsf{Can}_S(\cdot)$ whose image is isomorphic to $\mathbb{F}_2^n/S$ and is a subspace of $\mathbb{F}_2^n$. This improves on prior work, [CLLZ21a], which defined a set of canonical coset representatives that was not necessarily a group. The algebraic structure of $\mathsf{Can}_S(\cdot)$ allows us to prove more-sophisticated claims than what was possible with [CLLZ21a]'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 on a random coset state $|S_{\mathbf{x},\mathbf{z}}\rangle$ is independent of S, since S can be chosen *after* the adversary acts. This also forces a degree of independence from $\mathbf{x}$ and $\mathbf{z}$, since the choice of S influences those heavily.

Why monogamy-of-entanglement techniques fail. Prior works [CLLZ21a, 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.

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.

4.5 Obfuscation and Applications

We now describe how to obtain several applications with certified deletion from certified deletion of coset states, including our main application of obfuscation. As a first result, the certified deletion theorem where $\mathcal{Z}$ depends on $S^\perp + \mathbf{z}$ easily yields encryption with *publicly verifiable* certified deletion. The construction is essentially the same as in [BK23], but the more flexible $\mathcal{Z}$ allows additionally publishing an indistinguishability obfuscation of a membership program for $S^\perp + \mathbf{z}$. This idea can also be applied to the rest of the results in this section.

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

$$|S_{\mathbf{x},\mathbf{z}}\rangle\,, \mathsf{Obf}(P[S, T, \widetilde{\mathbf{x}}, C + \mathsf{Can}_S(\mathbf{x})])$$

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* [NY90], 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.

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.

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].[4]

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.

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

Chapter 5

Preliminaries

5.1 Indistinguishability Obfuscation and Differing Inputs Obfuscation

Here we define two notions of obfuscation: indistinguishability obfuscation (iO) and differing inputs obfuscation (diO). Indistinguishability obfuscation (iO) guarantees that for any two functionally equivalent circuits, their obfuscations are indistinguishable.

Definition 5.1.1 (Indistinguishability obfuscation)**.** *An indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a PPT algorithm iO 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,*

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

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

$$|\Pr[\mathsf{Adv}_\lambda(\mathsf{iO}(1^\lambda, C_{0,\lambda})) = 1] - \Pr[\mathsf{Adv}_\lambda(\mathsf{iO}(1^\lambda, C_{1,\lambda})) = 1]| = \mathsf{negl}(\lambda)(\lambda).$$

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 5.1.2 (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 Adv,*

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

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 $|\psi_{\mathsf{Adv}}\rangle$. 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 5.1.3 (Differing Inputs Obfuscation)**.** *An differing inputs obfuscator* diO *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, $\Pr[\tilde{C}(x) = C(x) \forall x : \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C)] = 1$
- ***Indistinguishability:*** *For all differing inputs circuit classes and all PPT distinguishers* D,

$$\left| \begin{array}{l} \Pr[D(C_0, C_1, \mathsf{aux}, \tilde{C}) = 1 : \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C_0)), (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}] \\ -\Pr[D(C_0, C_1, \mathsf{aux}, \tilde{C}) = 1 : \tilde{C} \leftarrow \mathsf{diO}(1^\lambda, C_1), (C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}] \end{array} \right| = \mathsf{negl}(\lambda)$$

If indistinguishability holds against QPT distinguishers, we say diO *is quantum-secure. If this holds and* diO *is a PPT algorithm, we instead say it is post-quantum.*

[BCP14] show in the classical setting 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_{Adv} and quantum advice $|\psi_{\mathsf{Adv}}\rangle$. 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 $|\psi_{\mathsf{Adv}}\rangle$, rendering it useless for subsequent experiments. However, if $|\psi_{\mathsf{Adv}}\rangle$ exists, then so does the state $|\psi_{\mathsf{Adv}}\rangle^{\otimes n}$, consisting of n copies of $|\psi_{\mathsf{Adv}}\rangle$. Since $|\psi_{\mathsf{Adv}}\rangle$ depends only on the circuit family (not the samples C_0, C_1), the copied state may be obtained non-uniformly.[1] Since aux is classical, it may be copied as well. Using one copy per distinguishing experiment allows the proof to succeed.

Concurrently to this work, Zhandry [Zha23] showed a stronger result that post-quantum iO implies post-quantum diO for a polynomial number of differing inputs even when the sampler may output quantum auxiliary input. This result is not necessary for our constructions, since we only require security against samplers outputting classical auxiliary input. We mention it here to give a more complete overview of differing inputs obfuscation.

Lemma 5.1.4 ([Zha23])**.** *Any indistinguishability obfuscator is a differing inputs obfuscator for families with a polynomial number of differing inputs, even if the associated distribution may output quantum auxiliary advice.*

We give a much simpler proof in the case of one differing input.

Lemma 5.1.5. *Any indistinguishability obfuscator is a differing inputs obfuscator for circuit families with one differing input, even if the associated distribution may output quantum auxiliary advice.*

Proof. Let $y^* \in \{0,1\}^n$ be the differing input. Say there is an adversary $\mathcal{A}_{\mathsf{diO}}$ which wins the differing inputs obfuscation game with probability noticeably greater than $1/2$, but which cannot break indistinguishability obfuscation. Then we will construct an "inner product" adversary $\mathcal{A}_{\mathsf{IP}}$ which guesses $\langle y^*, r\rangle$ with probability noticeably greater than $1/2$, over the choice of r. By [AC02], this implies the existence of an efficient adversary which finds y^* given just Π_0 and Π_1, violating the definition of a differing inputs circuits family. This adversary can be obtained uniformly from one copy of $\mathcal{A}_{\mathsf{IP}}$.

Fix any r and $b \in \{0,1\}$. Define the circuit $\Pi_{r,b}$ as follows:

$$\Pi_{r,b}(x) = \begin{cases} \Pi_0(x) & \text{if } \langle x, r\rangle = b \\ \Pi_1(x) & \text{otherwise} \end{cases}$$

[1] 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.

$\Pi_{r,b}$ is functionally equivalent to Π_0 if $\langle y^*, r\rangle = b$, and otherwise it is functionally equivalent to Π_1. To guess $\langle y^*, r\rangle$, run the differing inputs obfuscation distinguishing experiment with $\mathcal{A}_{\mathsf{diO}}$, using input $(\Pi_0, \Pi_1, \mathsf{iO}(\Pi_{r,b}))$, for a uniform bit b. Output $b \oplus b'$, where b' is $\mathcal{A}_{\mathsf{diO}}$'s output. There are two cases:

- Say $\langle y^*, r\rangle = 0$. Indistinguishability obfuscation implies that $\mathsf{iO}(\Pi_{r,0}) \approx \mathsf{iO}(\Pi_0)$ and $\mathsf{iO}(\Pi_{r,1}) \approx \mathsf{iO}(\Pi_1)$. In other words, $\mathsf{iO}(\Pi_{r,b}) \approx \mathsf{iO}(\Pi_b)$. By assumption, if $\mathcal{A}_{\mathsf{diO}}$ plays the experiment using $(\Pi_0, \Pi_1, \mathsf{iO}(\Pi_b))$, then it outputs b with probability noticeably greater than $1/2$. This is negligibly different from the original experiment.

- Say $\langle y^*, r\rangle = 1$. Similarly, indistinguishability obfuscation implies that $\mathsf{iO}(\Pi_{r,b}) \approx \mathsf{iO}(\Pi_{1\oplus b}))$. By assumption, if $\mathcal{A}_{\mathsf{diO}}$ plays the experiment using $(\Pi_0, \Pi_1, \mathsf{iO}(\Pi_{1\oplus b}))$, then it outputs $1 \oplus b$ with probability noticeably greater than $1/2$. This is negligibly different from the original experiment.

In both cases, this procedure predicts $\langle y^*, r\rangle$ with probability noticeably greater than $1/2$. □

5.2 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 iO 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 $\mathbf{s} \in \mathbb{F}_2^n$, and outputs 1 if $\mathbf{s} \in S$ and 0 otherwise. Second, when S is a subspace, we define a subspace-hiding obfuscator shO to be a program that obfuscates P_S. For security, we say that an adversary cannot distinguish $\mathsf{shO}(P_S)$ from $\mathsf{shO}(P_T)$, where T is a random superspace of S.

Definition 5.2.1 ([Zha19b], Def. 6.2)**.** *A subspace-hiding obfuscator* shO *for a field* $\mathbb{F}$ *and dimensions* d_S, d_T *is a PPT algorithm* shO *that takes a program* P_S *as input and outputs an obfuscated program* $\widetilde{P}$*.* shO *is secure if all QPT adversaries have negligible advantage in the following game:*

- *The adversary sends the challenger a subspace* $S \subset \mathbb{F}^\lambda$ *of dimension* d_S*.*

- *The challenger samples a random subspace* $T \subset \mathbb{F}^\lambda$ *of dimension* d_T *such that* $S \subseteq T$*. Then they sample* $b \leftarrow \{0,1\}$*, and if* $b = 0$ *they compute* $\widetilde{P} \leftarrow \mathsf{shO}(P_S)$*, and if* $b = 1$*, they compute* $\widetilde{P} \leftarrow \mathsf{shO}(P_T)$*. 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 5.2.2 ([Zha19b], Theorem 6.3)**.** *If injective one-way functions exist and* $|\mathbb{F}|^{n-d_T}$ *is exponential in* λ*, then any indistinguishability obfuscator, appropriately padded, is a subspace hiding obfuscator for field* $\mathbb{F}$ *and dimensions* d_S, d_T*.*

[CLLZ21a] extend subspace-hiding to also consider programs which decide membership in a coset of S.

Corollary 5.2.3 ([CLLZ21a])**.** *Let* iO *be an indistinguishability obfuscator and suppose that injective one-way functions exist, and consider the following game.*

- *The adversary sends the challenger a subspace* $S \subset \mathbb{F}_2^\lambda$ *of dimension* $\lambda/2$ *and a vector* $\mathbf{x} \in \mathbb{F}_2^\lambda$*.*

- *The challenger samples a random superspace $T \supset S$ of dimension $3\lambda/4$ and samples $\widetilde{\mathbf{x}} \leftarrow T + \mathbf{x}$. If $b = 0$, they compute $\widetilde{P} \leftarrow \mathsf{iO}(P_{S+\mathbf{x}})$, and if $b = 1$ they compute $\widetilde{P} \leftarrow \mathsf{iO}(P_{T+\widetilde{\mathbf{x}}})$. Then they send $\widetilde{P}$ to the adversary.*
- *The adversary makes a guess b' for b.*

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

5.3 Functional Encryption

Functional encryption allows a user holding a key for sk_f and a ciphertext $\mathsf{Enc}(x)$ to learn $f(x)$, but nothing more.

Definition 5.3.1 (Functional Encryption)**.** *A functional encryption scheme is associated with a class of functions $\mathcal{F}(\lambda)$ and a message space $\mathcal{M}_\lambda$. It consists of the following PPT algorithms:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter λ, then outputs a public key* pk *and a master secret key* msk.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$ *takes as input the master secret key* msk *and the description of a function $f \in \mathsf{F}$, then outputs a secret key sk_f.*
- $\mathsf{Enc}(\mathsf{pk}, m)$ *takes as input the public key* pk *and a message m, then outputs a ciphertext c.*
- $\mathsf{Dec}(\mathsf{sk}_f, c)$ *takes as input a secret key sk_f and a ciphertext encrypting a message $m \in \mathcal{M}_\lambda$, then outputs $f(m)$.*

Definition 5.3.2 (Decryption Correctness for FE)**.** *A functional encryption scheme must satisfy decryption correctness:*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{pk}, m)) \neq f(m) : \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{sk}_f, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)\end{array}\right] = \mathsf{negl}(\lambda)$$

Functional encryption security requires that an adversary cannot learn anything beyond the functions for which it has secret keys. This is formalized by requiring that they cannot distinguish between encryptions of two different messages which have the same evaluations under the functions which the adversary has keys for.

Definition 5.3.3 (FE Security)**.** *A functional encryption scheme, is (post-quantum) secure if the advantage of every QPT adversary in the following game is $\mathsf{negl}(\lambda)$:*

1. *The challenger samples $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pk to the adversary.*
2. *The following query phase is repeated a polynomial number of times:*
   (a) *The adversary adaptively submits a query $f_i \in \mathcal{F}(\lambda)$.*
   (b) *The challenger samples $(\mathsf{sk}_{f_i}, \mathsf{vk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$ and sends sk_{f_i} to the adversary.*
3. *The adversary sends two messages m_0 and m_1 to the challenger.*
4. *The challenger checks the deletion proofs. If $f_i(m_0) = f_i(m_1)$ for every f_i, then sample a random bit b and send $\mathsf{Enc}(\mathsf{pk}, m_b)$ to the adversary.*

5. *The following query phase is repeated a polynomial number of times:*
   (a) *The adversary adaptively submits a query* $f_i \in \mathcal{F}(\lambda)$.
   (b) *If* $f_i(m_0) = f_i(m_1)$*, the challenger samples* $(\mathsf{sk}_{f_i}, \mathsf{vk}_i)$ *and sends* sk_{f_i} *to the adversary.*
6. *The adversary outputs a bit* b' *and wins if* $b' = b$.

We say the functional encryption scheme has selective *security if this holds in the game where adversary must declare the challenge messages* m_0 *and* m_1 before *the challenger samples* $(\mathsf{pk}, \mathsf{sk})$.

Chapter 6

Coset State Framework

6.1 Coset Representatives

Since a coset $S + \mathbf{x}$ can be written many ways, e.g. as $S + \mathbf{x}'$ for any $\mathbf{x}' \in S + \mathbf{x}$, it is useful to identify it with a single, *canonical* element $\mathsf{Can}_S(\mathbf{x})$ from the set. Previously, [CLLZ21a] used the lexicographically first element of $S + \mathbf{x}$ as its canonical element. Although this is well-defined and efficient to compute, it is not particularly well-structured. As a more structured approach, we provide a canonical representation which is isomorphic to the quotient group $\mathbb{F}^n/S = \{S + x : x \in \mathbb{F}^n\}$. This provides useful properties such as homomorphism, e.g. $\mathsf{Can}_S(\mathbf{x}) + \mathsf{Can}_S(\mathbf{y}) = \mathsf{Can}_S(\mathbf{x} + \mathbf{y})$.

Canonical representations $\mathsf{Can}_S(\cdot)$ of vectors also give a canonical description for cosets $S + \mathbf{x}$. It consists of a description of S (e.g. a basis) and $\mathsf{Can}_S(\mathbf{x})$.

Complement-Defined Canonical Elements. Consider a subspace $S \subset T$. For example, T may be $\mathbb{F}^n$, a subspace of $\mathbb{F}^n$, or some other vector space. Most commonly, we will work with $T = \mathbb{F}_2^n$, a vector space over the binary field. We define the canonical representative of a coset $S + \mathbf{x}$ with respect to a complementary subspace of S, i.e. a subspace C such that $\mathsf{Span}(S, C) = T$ and $S \cap C = \{0\}$. Any $\mathbf{x} \in T$ can be uniquely written as

$$\mathbf{x} = \mathbf{x}_S + \mathbf{x}_C \qquad \text{where } \mathbf{x}_S \in S \text{ and } \mathbf{x}_C \in C \tag{6.1}$$

This decomposition implies that if $\mathbf{x}_C = \mathbf{y}_C$ for some $\mathbf{x}, \mathbf{y} \in T$, then $\mathbf{x} \in S + \mathbf{y}$, since in this case $\mathbf{x} = (\mathbf{x}_S - \mathbf{y}_S) + \mathbf{y}$ and $\mathbf{x}_S - \mathbf{y}_S \in S$.

Thus, we can define the canonical representative of $S + \mathbf{x}$ with respect to C as

$$\mathsf{Can}_{S,C}(\mathbf{x}) = \mathbf{x}_C \tag{6.2}$$

Canonical representatives can be efficiently computed using any basis $B = [B_S, B_C]$ of T that contains a basis B_S for S and a basis B_C for C. Given such a basis, compute the basis decomposition of $\mathbf{x}$ with respect to B and output the part corresponding to B_C.

Remark 6.1.1. *There many possible choice of the complementary space C. Furthermore, different choices of C lead to different canonical representatives. It would be convenient if we could take C to be the (unique) orthogonal complement of S. Unfortunately, orthogonal complements do not exist if T is defined over a finite field, since elements of a finite field cannot be ordered.*

To simplify notation, we consider the description of S to include the following information:

- A basis for S.
- A basis for a space C which is complementary to S.
- A basis for a space $C_\perp$ which is complementary to $S^\perp$.

Since the description of S determines complementary spaces for S and $S^\perp$, we may then simply write $\mathsf{Can}_S(\cdot) \coloneqq \mathsf{Can}_{S,C}(\cdot)$ and $\mathsf{Can}_{S^\perp}(\cdot) \coloneqq \mathsf{Can}_{S^\perp,C_\perp}(\cdot)$.

6.2 Coset Representative Properties

It should be immediately obvious that the set of canonical elements is precisely C. This observation gives us a good base to see several initial properties. First, the mapping $\mathsf{Can}_S : T \to C$ is homomorphic:

$$\mathsf{Can}_S(\mathbf{x}) + \mathsf{Can}_S(\mathbf{y}) = \mathsf{Can}_S(\mathbf{x} + \mathbf{y})$$

As an immediate consequence of the homomorphism, the set $C_T \coloneqq \{\mathsf{Can}_S(\mathbf{v}) : \mathbf{v} \in V\}$ forms a subspace for any subspace $V \subset T$. Second, since C is complementary to S, we know that $\mathsf{Dim}(C) = \mathsf{Dim}(T) - \mathsf{Dim}(S)$. Finally, it is isomorphic to the quotient group T/S.

Parity Balance. The homomorphism property implies a useful property for hiding information; the set of canonical representatives is balanced with respect to dot product with a random vector with very high probability. A similar property holds for random C and fixed $\mathbf{r} \neq \vec{0}$, e.g. $\mathbf{r} = \vec{1}$, though we omit the proof of that.

Lemma 6.2.1. *Let $n \in \mathbb{N}$. Let S and C be complementary subspaces of $\mathbb{F}_2^n$. Let $T \supset S$ be a superspace of S. For any $\mathbf{r} \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2$, let*

$$D_{T,\mathbf{r},b} = \{\mathsf{Can}_S(\mathbf{x}) : \mathbf{x} \in T \text{ and } \mathbf{x} \cdot \mathbf{r} = b\}$$

Then

$$\Pr_{\mathbf{r} \leftarrow \mathbb{F}_2^\lambda}[|D_{T,\mathbf{r},0}| = |D_{T,\mathbf{r},1}|] = 1 - 2^{-(\mathsf{Dim}(T) - \mathsf{Dim}(S))}$$

Proof. Since $\mathsf{Can}_S : \mathbb{F}_2^n \to C$ is a homomorphism and T is a subspace of $\mathbb{F}_2^n$, the set $C_T \coloneqq \{\mathsf{Can}_S(\mathbf{t}) : \mathbf{t} \in T\}$ forms a subspace. The dimension of C_T is the dimension of T minus the dimension of the kernel of Can_S restricted to inputs $t \in T$. This is precisely S since $S \subset T$. So $\dim(C_T) = \dim(T) - \dim(S)$.

If $\mathbf{r} \notin C_T^\perp$, then there exists $\mathbf{x} \in C_T^\perp$ such that $r \cdot x = 1$. Addition by $\mathbf{x}$ gives a bijection between $D_{T,r,0}$ and $D_{T,r,1}$. The probability of sampling such an $\mathbf{r}$ is $1 - |C_T^\perp|/|\mathbb{F}_2^n|$. Since $|C_T^\perp| = 2^{n-\dim(C_T)} = 2^{n-\dim(T)+\dim(S)}$, we have the result.

□

Embedding. A subspace T can be embedded in a larger subspace by adjoining it with a set of linearly independent vectors: $T \subset \mathsf{Span}(T, B)$. For example, we can extend $\mathbb{F}^n$ to $\mathbb{F}^{2n}$ in this manner. $\mathsf{Can}_S(\cdot)$ is compatible with such extensions, in the sense that the extension does not modify the canonical representatives of any element from the original subspace.

As a result, we can naturally obtain canonical representatives for subspaces $S \subset T \subset W$ where the set of canonical elements for S is a subset of the canonical elements for T.

Lemma 6.2.2 (Embedding Friendliness)**.** *Let T and C_T be complementary subspaces of some vector space W. Let S and C_S be complementary subspaces of T. Let $C = \mathsf{Span}(C_S, C_T)$.*

For all such W, T, S, C and for all $\mathbf{c}_T \in C_T$ and $\mathbf{x} \in T$,

$$\mathsf{Can}_{S,C}(\mathbf{x} + \mathbf{c}_T) = \mathsf{Can}_{S,C_S}(\mathbf{x}) + c_T$$

Proof. $\mathsf{Can}_{S,C}(\mathbf{x} + \mathbf{c}_T) \in C$ is the unique element such that

$$\mathbf{x} + \mathbf{c}_T = \mathbf{x}_S + \mathsf{Can}_{S,C}(\mathbf{x} + \mathbf{c}_T)$$

for some $\mathbf{x}_S \in S$. C_S and C_T are complementary subspaces of C since $C_T \cap T = \{\mathbf{0}\}$ and $C_S \subset T$. So we can also uniquely decompose $\mathsf{Can}_{S,C}(\mathbf{x} + \mathbf{c}_T)$ into $\mathbf{c}_S + \mathbf{c}'_T$ where $\mathbf{c}_S \in C_S$ and $\mathbf{c}'_T \in C_T$. In other words,

$$\mathbf{x} + \mathbf{c}_T = \mathbf{x}_S + (\mathbf{c}_S + \mathbf{c}'_T)$$

Observe that $\mathbf{x} + \mathbf{c}_T$ is a decomposition of $\mathbf{x} + \mathbf{c}_T$ into elements of T and $\mathcal{C}_T$, so it must be unique. Since $(\mathbf{x}_S + \mathbf{c}_S) \in T$ and $\mathbf{c}'_T \in C_T$, these must match that decomposition, i.e. $(\mathbf{x}_S + \mathbf{c}_S) = \mathbf{x}$ and $\mathbf{c}'_T = \mathbf{c}_T$. Finally, $\mathbf{c}_S$ exactly matches the definition of $\mathsf{Can}_{S,C}(\mathbf{x})$. □

As an example application, consider extending $\mathbb{F}^n$ to $\mathbb{F}^{2n}$ by appending some 0s and adjoining it with $C_T = \mathsf{Span}(e_i : i \in [n+1, 2n])$. The embedding lemma says that if we consider any $\mathbf{x}_1 \| \mathbf{x}_2 = \mathbf{x}_1 \| \mathbf{0}^n + \mathbf{0}^n \| \mathbf{x}_2$ for $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}_2^n$, then the canonical representative $\mathsf{Can}_S(\mathbf{x}_1 \| \mathbf{x}_2)$ is $\mathsf{Can}_S(\mathbf{x}_1) \| \mathbf{0}^n + \mathbf{0}^n \| \mathbf{x}_2$ for all $S \subset \mathbb{F}^n$.

Rerandomization. A useful tool for subspaces is the ability to perform a change of basis. For a random change of basis M, this rerandomizes the subspace. $\mathsf{Can}_S(\cdot)$ behaves in a very simple manner under changes of basis.

Lemma 6.2.3 (Canonical Rerandomization)**.** *Let S and C be complementary subspaces of T. Then for all invertible linear transformations $M : T \to T$ and for all $\mathbf{x} \in T$,*

$$\mathsf{Can}_{MS,MC}(M\mathbf{x}) = M\mathsf{Can}_{S,C}(\mathbf{x})$$

We briefly note that because $M : T \to T$ is an invertible linear transformation, MS and MC are also complementary subspaces of T, so $\mathsf{Can}_{MS,MC}(\cdot)$ is well-defined.

Proof. $\mathsf{Can}_{MS,MC}(M\mathbf{x})$ is the unique value $\mathbf{x}_{MC} \in MC$ such that

$$M\mathbf{x} = \mathbf{x}_{MS} + \mathbf{x}_{MC}$$

for some $\mathbf{x}_{MS} \in MS$. By definition of MC and MS, it must be the case that $\mathbf{x}_{MC} = M\mathbf{x}_C$ for some $\mathbf{x}_C \in C$ and $\mathbf{x}_{MS} = M\mathbf{x}_S$ for some $\mathbf{x}_S \in S$. Now M's invertibility implies that $\mathbf{x} = \mathbf{x}_S + \mathbf{x}_C$. This is a decomposition of $\mathbf{x}$ into S and C, so $\mathbf{x}_C = \mathsf{Can}_{S,C}(\mathbf{x})$. □

We refer to $\mathsf{Can}_{MS,MC}(M\mathbf{x})$ as the canonical rerandomization of $\mathsf{Can}_{S,C}(\mathbf{x})$ because M is applied to the whole space T to rerandomize it. Thus we should consider mapping $S \mapsto MS$ and $C \mapsto MC$, along with $x \mapsto Mx$ for any $x \in T$. Since we consider the description of S to also include a complementary space C, we overload notation slightly by writing $\mathsf{Can}_{MS}(\cdot)$ to mean $\mathsf{Can}_{MS,MC}(\cdot)$.

6.3 Delayed Preparation of Coset States

A useful tool for analyzing random BB84 states is the ability to delay the choice of basis. The preparer can send a random BB84 state by preparing an EPR pair, then sending one half of it to the receiver. Later, the preparer can decide if the sent state should be in the computational basis ($|0\rangle$ or $|1\rangle$) or the Hadamard basis ($|+\rangle$ or $|-\rangle$) and measure their own half of the EPR pair to determine the state. By deciding the basis later, we can argue that any adversarial action on the BB84 state $H^\theta |x\rangle$ is independent of the chosen basis.

We show that it is possible to do an analogous technique for preparing random coset states. The preparer constructs some number of EPR pairs and sends one half of each pair to the receiver. Later, it chooses the subspace S and measures its own halves to determine the coset offsets.

Let $\mathsf{Can}_S(\mathbf{x})$ be a function which computes the canonical representative of the coset $S + \mathbf{x} \subset \mathbb{F}^n$.

1. The preparer constructs n EPR pairs $1/\sqrt{|\mathbb{F}|^n} \sum_{\mathbf{v}\in\mathbb{F}^n} |\mathbf{v}\rangle_{\mathcal{A}} \otimes |\mathbf{v}\rangle_{\mathcal{B}}$.
2. The preparer sends register $\mathcal{B}$ to the receiver.
3. The preparer chooses $S \subset \mathbb{F}^n$.
4. The preparer computes $\mathsf{Can}_{S^\perp}(\cdot)$ in the Hadamard basis on register $\mathcal{A}$, then measures the result. Call the result $\mathbf{z}$.
5. The preparer computes $\mathsf{Can}_S(\cdot)$ on register $\mathcal{A}$ in the computational basis, then measures the result. Call the result $\mathbf{x}$.

Figure 6.1: Protocol for Delaying the Preparation of an n-bit Coset State

Lemma 6.3.1. *At the end of the protocol in Figure 6.1, register $\mathcal{B}$ holds the coset state $|S_{x,z}\rangle$, where x and z are the preparer's measurement results.*

Proof. Let $\mathsf{Can}_{S^\perp} = \{\mathsf{Can}_{S^\perp}(\mathbf{z}) : \mathbf{z} \in \mathbb{F}^n\}$. Every element $\mathbf{v} \in \mathbb{F}^n$ can be uniquely written as $\mathbf{v} = \mathbf{s} + \mathbf{z}$ for some $\mathbf{s} \in S^\perp$ and $\mathbf{z} \in \{\mathsf{Can}_{S^\perp}\}$. Therefore the purified state after step 4 is

$$\frac{1}{\sqrt{|\mathbb{F}|^n}} \sum_{\mathbf{x}\in\{\mathsf{Can}_S\}} |\mathbf{x}\rangle_{\mathcal{X}} \otimes H^{\otimes 2n} \left(\sum_{\mathbf{s}\in S^\perp} |\mathbf{s}+\mathbf{x}\rangle_{\mathcal{A}} \otimes |\mathbf{s}+\mathbf{x}\rangle_{\mathcal{B}} \right)$$

Writing register $\mathcal{A}$ in the computational basis yields

$$\frac{1}{|\mathbb{F}|^n} \sum_{\mathbf{z}\in\{\mathsf{Can}_S\}} |\mathbf{z}\rangle_{\mathcal{Z}} \otimes (I^{\otimes n} \otimes H^{\otimes n}) \left(\sum_{\mathbf{s}\in S^\perp} \sum_{\mathbf{x}\in\mathbb{F}^n} \omega_{|\mathbb{F}|}^{\mathbf{x}\cdot(\mathbf{s}+\mathbf{z})} |\mathbf{x}\rangle_{\mathcal{A}} \otimes |\mathbf{s}+\mathbf{z}\rangle_{\mathcal{B}} \right)$$

By grouping the terms dependent on $\mathbf{s} \in S$ separately from those dependent only on $\mathbf{z}$ and $\mathbf{x}$, we see that

the state is

$$\frac{1}{|\mathbb{F}|^n}\sum_{\substack{\mathbf{z}\in\{\mathsf{Can}_S\}\\ \mathbf{x}\in\mathbb{F}^n}}\omega_{|\mathbb{F}|}^{\mathbf{z}\cdot\mathbf{x}}\left|\mathbf{z}\right\rangle_{\mathcal{Z}}\otimes\left|\mathbf{x}\right\rangle_{\mathcal{A}}\otimes\left(H^{\otimes n}\sum_{\mathbf{s}\in S^{\perp}}\omega_{|\mathbb{F}|}^{\mathbf{x}\cdot\mathbf{s}}\left|\mathbf{s}+\mathbf{z}\right\rangle_{\mathcal{B}}\right)$$
$$=\frac{\sqrt{|S|}}{|\mathbb{F}|^n}\sum_{\substack{\mathbf{z}\in\{\mathsf{Can}_S\}\\ \mathbf{x}\in\mathbb{F}^n}}\left(\omega_{|\mathbb{F}|}^{\mathbf{z}\cdot\mathbf{x}}\left|\mathbf{z}\right\rangle_{\mathcal{Z}}\otimes\left|\mathbf{x}\right\rangle_{\mathcal{A}}\right)\otimes\left|S_{\mathbf{x},\mathbf{z}}\right\rangle_{\mathcal{B}}$$

Tracing out registers $\mathcal{Z}$ and $\mathcal{A}$ completes the proof. □

Chapter 7

Certified Deletion for Coset States

Previously, [BK23] gave a general certified deletion theorem for BB84 states: $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. Roughly, their theorem allowed for encoding a single bit b in a quantum state $H^\theta |x\rangle$. If an adversary holding an encryption of θ guesses the values of x which were encoded in the Hadamard basis, then b becomes information-theoretically lost – despite being previously information-theoretically determined.

We show that if we instead base the deletable quantum state on *coset* states, then it is safe to reveal additional information about the underlying cosets. This information can be used to provide repeated partial access to the encoded data, *without compromising the complete destruction of the data upon producing a valid deletion certificate*.

As a base, we can encode a single bit b using a subspace $S \subset \mathbb{F}_2^\lambda$ along with three vectors $x, z, r \in \mathbb{F}_2^\lambda$ as

$$|S_{x,z}\rangle,\ r,\ (\mathsf{Can}_{S,C}(x) \cdot r) \oplus b \tag{7.1}$$

We also want to include some side-information about $S + \mathbf{x}$ and $S^\perp + \mathbf{z}$ to allow controlled access to the encoded bit. So instead, we will consider outputting $|S_{x,z}\rangle$ along with a sample from a distribution

$$\mathcal{Z}(S, S + x,\ S^\perp + z,\ r,\ (\mathsf{Can}_{S,C}(x) \cdot r) \oplus b) \tag{7.2}$$

which hides S, x, and z in some way. In [BK23], $\mathcal{Z}$ is required to be semantically hiding, i.e. $\mathcal{Z}(S) \approx \mathcal{Z}(0)$. We will relax the requirement to instead satisfy a weaker notion called "coset-hiding". Roughly, coset-hiding requires that it is indistinguishable whether $\mathcal{Z}$ uses the original coset $S + \mathbf{x}$, or $T + \mathbf{x}$ for a random superspace $T \supset S$.[1]

Definition 7.0.1 (Coset-Hiding)**.** *A distribution $\mathcal{D}(\cdot)$ is* (d_S, d_T, n)-coset-hiding *if there exists a simulator* Sim *such that for all subspaces $S \subset \mathbb{F}_2^n$ with dimension d_S and all cosets $S + \mathbf{x}$ of S,*

$$\mathcal{D}(S + \mathbf{x}) \approx_c \left\{ \mathsf{Sim}(T + \widetilde{\mathbf{x}}) : \begin{array}{c} T \leftarrow \mathsf{Supspace}_{d_T}(S) \\ \widetilde{\mathbf{x}} = \mathsf{Can}_T(\mathbf{x}) \end{array} \right\}$$

To formalize the certified deletion guarantees of this encoding, we will consider an experiment where the adversary receives the encoding of a bit b along with side information $\mathcal{Z}$, then produces the deletion certificate. If the certificate is valid, then the experiment outputs the adversary's view; otherwise it outputs $\perp$.

[1] When we write $\mathcal{Z}(S + \mathbf{x})$, we consider $\mathcal{Z}$ to receive a description of the coset, which consists of description for S (see Chapter 6) and the *canonical* element of $S + \mathbf{x}$. Frequently, we will explicitly write $S + \mathsf{Can}_S(\mathbf{x})$ to emphasize that the description actually uses the canonical element.

Definition 7.0.2 (Certified Deletion Game for Cosets)**.** *The* $\mathsf{Co\text{-}CD}^{n,d_s}_{\mathcal{A},\mathcal{Z}}(b)$ *game is parameterized by a bit* b*, dimensions* n *and* d_s*, a quantum algorithm* $\mathcal{A}$*, and a distribution* $\mathcal{Z}$*. It is played as follows:*

1. ***Challenge:*** *Sample a* d_S*-dimensional subspace* $S \subset \mathbb{F}_2^n$*. Sample* $\mathbf{x}', \mathbf{z}', \mathbf{r}' \leftarrow \mathbb{F}_2^n$*. Let* $\widetilde{b} = (\mathsf{Can}_S(\mathbf{x}) \cdot \mathbf{r}) \oplus b$*. Initialize the adversary with*

$$|S_{\mathbf{x},\mathbf{z}}\rangle \quad \textit{and} \quad \mathcal{Z}(S,\ S+\mathbf{x},\ S^\perp+\mathbf{z},\ \mathbf{r},\ \widetilde{b})$$

2. ***Response:*** *The adversary sends back a certificate* cert *and a register* $\mathcal{R}$*.*

3. ***Output:*** *If* $\mathsf{cert} \in S^\perp + \mathbf{z}$*, output* $\mathcal{R}$*. Otherwise output* $\perp$*.*

$\mathcal{Z}$ is defined in a general manner in the Co-CD experiment – it is allowed to access all of the information about $|S_{x,z}\rangle$. This allows for more flexibility in precisely which pieces of information $\mathcal{Z}$ depends upon. We can write $\mathcal{Z} = \mathcal{Z}_1 \circ \mathcal{Z}_2$ as a composition of a computationally hiding distribution $\mathcal{Z}_1$ with a statistically hiding distribution $\mathcal{Z}_2$ which controls precisely what information $\mathcal{Z}_1$ can depend on. This allows a clean separation of what information must be statistically hidden, and what can be computationally hidden.

As the first part of our coset techniques, we consider a $\mathcal{Z}_2$ that removes $S + \mathbf{x}$ from its output. We show that if $\mathcal{Z}_1(S, S^\perp + \mathbf{z}, \dots)$ semantically hides S and subspace-hides $S^\perp + \mathbf{z}$, then the Co-CD game yields information-theoretic security. This theorem gives a way to safely publish a verification key by obfuscating a membership program for $S^\perp + \mathbf{z}$.

Theorem 7.0.3. *Let* $\gamma = \omega(\log(\lambda))$*,* $d_{S^\perp} = n - d_S$ *for* $d_S \in \mathbb{N}$*, and* $n \in \mathbb{N}$ *with* $n > d_S + \gamma$*. Let* $\mathcal{Z} = \mathcal{Z}_1 \circ \mathcal{Z}_2$ *where* $\mathcal{Z}_2(S, S+\mathbf{x}, S^\perp+\mathbf{z}, \mathbf{r}, \widetilde{b})$ *outputs* $(S, S^\perp+\mathbf{z}, \mathbf{r}, \widetilde{b})$*. If for all* $S, S^\perp + \mathbf{z} \subset \mathbb{F}_2^n$*, all* $\mathbf{r} \in \mathbb{F}_2^n$*, and all* $\widetilde{b}$*,*

- $\mathcal{Z}_1(\cdot,\ S^\perp+\mathbf{z},\ \mathbf{r},\ \widetilde{b})$ *is semantically hiding*
- *and* $\mathcal{Z}_1(S,\ \cdot,\ \mathbf{r},\ \widetilde{b})$ *is* $(d_{S^\perp}, d_{S^\perp}+\gamma, n)$*-coset-hiding,*

then for all QPT $\mathcal{A}$*,*

$$\mathsf{TD}[\mathsf{Co\text{-}CD}^{d_S,n}_{\mathcal{A},\mathcal{Z}}(0), \mathsf{Co\text{-}CD}^{d_S,n}_{\mathcal{A},\mathcal{Z}}(1)] = \mathsf{negl}(\lambda)$$

As example parameters, consider $n = \lambda$, $d_S = \lambda/2$, and $\gamma = \lambda/4$. We prove this theorem in Section 7.1.

Next, we extend the result so that $\mathcal{Z}_1$ can slightly depend on $S + x$. Specifically, we will consider giving $\mathcal{Z}$ a random super-coset $T + \widetilde{\mathbf{x}} \supset S + \mathbf{x}$. The fact that $\mathbf{x}$ is not used directly in $\mathcal{Z}_1$ is quite important, because this prevents an adversary from ignoring $|S_{\mathbf{x},\mathbf{z}}\rangle$ and directly breaking $\mathcal{Z}_1$ to obtain the encoded bit.

Theorem 7.0.4. *Let* $\gamma = \omega(\log(\lambda))$*,* $d_{S^\perp} = n - d_S$ *for* $d_S \in \mathbb{N}$*, and* $n \in \mathbb{N}$ *with* $n > d_S + \gamma$*. Let* $\mathcal{Z} = \mathcal{Z}_1 \circ \mathcal{Z}_2$ *where* $\mathcal{Z}_2(S, S+\mathbf{x}, S^\perp+\mathbf{z}, \mathbf{r}, \widetilde{b})$ *samples* $T \leftarrow \mathsf{Supspace}_{\mathsf{d_S}+\gamma}(S)$*, computes* $\widetilde{\mathbf{x}} = \mathsf{Can}_T(\mathbf{x})$*, then outputs* $(S, T+\widetilde{\mathbf{x}}, S^\perp+\mathbf{z}, \mathbf{r}, \widetilde{b})$*.*

If for all $S, S+\mathbf{x}, S^\perp+\mathbf{z} \subset \mathbb{F}_2^n$*, all* $\mathbf{r} \in \mathbb{F}_2^n$*, and all bits* $\widetilde{b}$*, with overwhelming probability over* $T \leftarrow \mathsf{Supspace}_{d_S+\gamma}(S)$*,*

- $\mathcal{Z}_1(\cdot,\ T+\widetilde{\mathbf{x}},\ S^\perp+\mathbf{z},\ \mathbf{r}, \widetilde{b})$ *is semantically hiding*
- *and* $\mathcal{Z}_1(S,\ T+\widetilde{\mathbf{x}},\ \cdot,\ \mathbf{r}, \widetilde{b})$ *is* $(d_{S^\perp}, d_{S^\perp}+\gamma, n)$*-coset-hiding,*

then for all QPT $\mathcal{A}$,

$$\mathsf{TD}[\mathsf{Co\text{-}CD}^{d_S,n}_{\mathcal{A},\mathcal{Z}}(0), \mathsf{Co\text{-}CD}_{\mathcal{A},\mathcal{Z}}(1)] = \mathsf{negl}(\lambda)$$

As example parameters, consider $n = \lambda$, $d_S = \lambda/2$, and $\gamma = \lambda/4$. We prove Theorem 7.0.4 by invoking Theorem 7.0.3 on a smaller space $\mathbb{F}_2^{d_S+\gamma}$ then embedding that space into $\mathbb{F}_2^n$ and rerandomizing the variables used in the experiment. See Section 7.2 for the proof.

Remark 7.0.5. *It is also possible to directly prove Theorem 7.0.4 in a similar manner to Theorem 7.0.3. However, splitting the two theorems provides some additional insight into the structures required for deletion. For instance, it is* also *possible to prove a similar result to Theorem 7.0.4 by reducing to [BK23]'s certified deletion theorem. Despite the fact that their theorem does not allow any information leakage about S, $\mathbf{x}$, or $\mathbf{z}$, a similar embedding argument still shows that $T + \widetilde{x}$ can be leaked in the side-information.*

7.1 Proof with Dual Coset Leakage

Proof of Theorem 7.0.3. Consider the following hybrids, where $\mathsf{Hyb}_0(b) = \mathsf{Co\text{-}CD}_{\mathcal{A},\mathcal{Z}}(b)$.

- $\mathsf{Hyb}_1(b)$: We purify the generation of $|S_{\mathbf{x},\mathbf{z}}\rangle$ according to the procedure in Figure 6.1, which we recall here:

  1. Prepare n EPR pairs $\propto \sum_{\mathbf{x}\in\mathbb{F}_2^n} |x\rangle_{\mathcal{R}_\mathcal{C}} \otimes |x\rangle_{\mathcal{R}_\mathcal{A}}$.
  2. Measure $\mathbf{z} \leftarrow \mathsf{Can}_{S^\perp}(H^{\otimes n}\mathcal{R}_\mathcal{C})$ in the Hadamard basis and $\mathbf{x} \leftarrow \mathsf{Can}_S(\mathcal{R}_\mathcal{C})$ in the computational basis.
  3. Output $\mathcal{R}_\mathcal{A}$ as the prepared coset state.

  Using $\mathcal{R}_\mathcal{A}$ in place of $|S_{\mathbf{x},\mathbf{z}}\rangle$, proceed as in Hyb_0. Explicitly, the adversary is now initialized with

  $$\text{register } \mathcal{R}_\mathcal{A} \quad \text{and} \quad \mathcal{Z}_1(S,\ S^\perp + \mathbf{z},\ \mathbf{r},\ \widetilde{b})$$

- $\mathsf{Hyb}_2(b)$: This is $\mathsf{Hyb}_1(b)$ with the following change. Instead of computing $\widetilde{b}$ as $b \oplus (\mathbf{r}\cdot\mathsf{Can}_S(\mathbf{x}))$, sample $\widetilde{b} \leftarrow \mathbb{F}_2$ uniformly at random. Then, at the *end* of the **output** phase, if $\widetilde{b} \neq b \oplus (\mathbf{r}\cdot\mathsf{Can}_S(\mathbf{x}))$, abort and output $\perp$. Otherwise output as usual.

- $\mathsf{Hyb}_3(b)$: The only change from $\mathsf{Hyb}_2(b)$ is as follows. Instead of measuring $\mathbf{x} \leftarrow \mathsf{Can}_S(\mathcal{R}_\mathcal{C})$ in the computational basis while preparing register $\mathcal{R}_\mathcal{A}$, delay this measurement until the end of the **output** phase just before comparing $\widetilde{b}$ to $b \oplus (\mathbf{r}\cdot\mathsf{Can}_S(\mathbf{x}))$.

- $\mathsf{Hyb}_4(b)$: The only change from $\mathsf{Hyb}_3(b)$ is as follows. After checking whether $\mathsf{cert} \in S^\perp + \mathbf{z}$ in the **output** phase, perform an additional measurement $\{\Pi_{\mathsf{cert}}, I - \Pi_{\mathsf{cert}}\}$ on register $\mathcal{R}_\mathcal{C}$, where

  $$\Pi_{\mathsf{cert}} := H^{\otimes n} |\mathsf{cert}\rangle \langle \mathsf{cert}| H^{\otimes n}$$

  If the result is Π_{cert}, continue the **output** phase as in $\mathsf{Hyb}_3(b)$. Otherwise, immediately output $\perp$.

  Note that the projection is done before measuring $\mathbf{x} \leftarrow \mathsf{Can}_S(\mathcal{R}_\mathcal{C})$ and checking whether $\widetilde{b}$ matches $b \oplus (\mathbf{r}\cdot\mathsf{Can}_S(\mathbf{x}))$.

We begin by showing that the distance between $b = 0$ and $b = 1$ in the last hybrid is negligible.

Claim 7.1.1. $\mathsf{TD}[\mathsf{Hyb}_4(0), \mathsf{Hyb}_4(1)] = \mathsf{negl}(\lambda)$

Proof. The probability of the experiment outputting $\perp$ before the final check of $\widetilde{b}$ is independent of b. Additionally, the trace distance conditioned on outputting $\perp$ is 0. Therefore the overall trace distance is no more than the trace distance conditioned on the experiment not outputting $\perp$ before it checks $\widetilde{b}$.

If this occurs, then $\mathbf{x}$ is the result of a computational basis measurement done on a Hadamard basis state $\Pi_{\mathsf{cert}} := H^{\otimes n}$. In particular,

$$\Pr_{\mathsf{Hyb}_4(b)}[\widetilde{b} = b \oplus (\mathbf{r} \cdot \mathsf{Can}_S(\mathbf{x}))] = \Pr_{\mathbf{x} \leftarrow \mathbb{F}_2^n}[\widetilde{b} \oplus b = (\mathbf{r} \cdot \mathsf{Can}_S(\mathbf{x}))]$$

If $\mathbf{r} \neq \mathbf{0}^n$, which occurs with $1 - 2^{-n}$ probability, this probability is precisely $1/2$, independently of b. Since this check is the only place in the experiment where b is used, the claim follows. □

Next, we show that the distance between $b = 0$ and $b = 1$ is negligible in each prior hybrid by relating Hyb_{i+1} to Hyb_i.

Claim 7.1.2. $\mathsf{TD}[\mathsf{Hyb}_3(0), \mathsf{Hyb}_3(1)] = \mathsf{negl}(\lambda)$

We defer the proof of Claim 7.1.2 until the end of the proof for easier reading, since it is much more involved than the other claims.

Claim 7.1.3. $\mathsf{TD}[\mathsf{Hyb}_2(0), \mathsf{Hyb}_2(1)] = \mathsf{negl}(\lambda)$

Proof. Register $\mathcal{R}_{\mathcal{C}}$ is disjoint from the adversary's view and checking $\mathsf{cert} \in S^{\perp} + \mathbf{z}$ uses the already-measured $\mathbf{z}$. Thus, measurements on it can be commuted past the adversary's actions in the **response** phase and the certificate check in the **output** phase. As a result, $\mathsf{Hyb}_2(b) = \mathsf{Hyb}_3(b)$ and the claim follows from Claim 7.1.2. □

Claim 7.1.4. $\mathsf{TD}[\mathsf{Hyb}_1(0), \mathsf{Hyb}_1(1)] = \mathsf{negl}(\lambda)$

Proof.

$$\mathsf{TD}[\mathsf{Hyb}_1(0), \mathsf{Hyb}_1(1)] \leq 2\mathsf{TD}[\mathsf{Hyb}_2(0), \mathsf{Hyb}_2(1)]$$

since $\mathsf{Hyb}_2(b)$ is identically distributed to the distribution which outputs $\perp$ with probability $1/2$ and otherwise outputs $\mathsf{Hyb}_1(b)$. $2\mathsf{negl}(\lambda)$ is still negligible, so Claim 7.1.3 implies the claim. □

Claim 7.1.5. $\mathsf{TD}[\mathsf{Hyb}_0(0), \mathsf{Hyb}_0(1)] = \mathsf{negl}(\lambda)$

Proof. $\mathsf{Hyb}_0(b) = \mathsf{Hyb}_1(b)$ since the purified procedure precisely creates the state $|S_{\mathbf{x},\mathbf{z}}\rangle$ in register $\mathcal{R}_{\mathcal{A}}$ (Lemma 6.3.1). Therefore Claim 7.1.4 implies the claim. □

To complete the proof, we fill in the missing proof of Claim 7.1.2.

Proof of Claim 7.1.2. The only difference between $\mathsf{Hyb}_3(b)$ and $\mathsf{Hyb}_4(b)$ is the addition of the measurement $\{\Pi_{\mathsf{cert}}, I - \Pi_{\mathsf{cert}}\}$ and post-selection on outcome Π_{cert}. Therefore if the experiment aborts due to obtaining result $I - \Pi_{\mathsf{cert}}$ with only negligible probability, the gentle measurement lemma (Lemma 2.3.1) and Claim 7.1.1 together imply the claim.

We show that this is the case. Consider the following hybrid experiments:

- $\mathsf{Hyb}_{3,0}(b)$ is $\mathsf{Hyb}_3(b)$, except it stops immediately after applying $\{\Pi_{\mathsf{cert}}, I - \Pi_{\mathsf{cert}}\}$ (if this point is reached) and outputs an error message 1 if the experiment would abort due to this check. Otherwise it outputs 0.

- $\mathsf{Hyb}_{3,1}(b)$ is the same as $\mathsf{Hyb}_3(b)$, except it replaces $\mathcal{Z}(S, S^\perp + \mathbf{z}, \mathbf{r}, \widetilde{b})$. Specifically, instead of sampling S directly, first sample a random subspace $R^\perp \subset \mathbb{F}_2^n$ of dimension $d_{S^\perp} + \gamma$ and sample $S^\perp \subset R^\perp$ as a random subspace. Then, initialize the adversary with $\mathsf{Sim}(R^\perp + \widetilde{\mathbf{z}}, \mathbf{r}, \widetilde{b})$ where $\widetilde{\mathbf{z}} = \mathsf{Can}_{R^\perp}(\mathbf{z})$.
- $\mathsf{Hyb}_{3,2}(b)$ is the same as $\mathsf{Hyb}_{3,1}(b)$, except that it performs an additional measurement $\widetilde{\mathbf{z}} \leftarrow \mathsf{Can}_{R^\perp}(H^{\otimes n}\mathcal{R}_\mathcal{C})$ while preparing $|S_{\mathbf{x},\mathbf{z}}\rangle$. It uses this $\widetilde{\mathbf{z}}$ to prepare $\mathsf{Sim}(R^\perp + \widetilde{\mathbf{z}}, \mathbf{r}, \widetilde{b})$.
- $\mathsf{Hyb}_{3,3}(b)$ is the same as $\mathsf{Hyb}_{3,2}(b)$, except we commute the measurement of $\mathbf{z}$ past the measurement of $\widetilde{\mathbf{z}}$ and the adversary's operations in the **response** phase. Additionally, since $S^\perp$ is no longer used until after the **response** phase (only $R^\perp$ is required before then), also delay the sampling of $S^\perp$.
- $\mathsf{Hyb}_{3,4}(b)$ is the same as $\mathsf{Hyb}_{3,3}(b)$, but instead of measuring $\mathbf{z} \leftarrow \mathsf{Can}_{S^\perp}(H^{\otimes n}\mathcal{R}_\mathcal{C})$ and checking that $\mathsf{cert} \in S^\perp + \mathbf{z}$, the challenger measures whether the contents of $\mathcal{R}_\mathcal{C}$ belong to $S^\perp + \mathsf{cert}$ in the Hadamard basis and accepts the certificate if so.

The outcome of $\mathsf{Hyb}_{3,1}(b)$ is computationally indistinguishable from $\mathsf{Hyb}_{3,0}(b)$ by the hiding of $\mathcal{Z}_1$, since the rest of the hybrid (including the measurement) requires only information about S, $S^\perp + \mathbf{z}$, $\mathbf{r}$, and $\widetilde{b}$ (but not $R^\perp$). The outcome of $\mathsf{Hyb}_{3,2}(b)$ is identical to $\mathsf{Hyb}_{3,1}(b)$ since $\mathcal{R}_\mathcal{C}$ is supported on vectors in $S^\perp + \mathbf{z} \subset R^\perp + \mathbf{z}$ and $\mathsf{Can}_{R^\perp}(\cdot)$ is deterministic on vectors in $R^\perp + \mathbf{z}$. The outcome of $\mathsf{Hyb}_{3,3}(b)$ is identical to $\mathsf{Hyb}_{3,2}(b)$ since the measurements $\mathsf{Can}_{S^\perp}(H^{\otimes n}\mathcal{R}_\mathcal{C})$ and $\mathsf{Can}_{R^\perp}(H^{\otimes n}\mathcal{R}_\mathcal{C})$ can be simultaneously diagonalized (using the Hadamard basis), so they commute, and $\mathcal{R}_\mathcal{C}$ is disjoint from the adversary's operations in the **response** phase. The outcome of $\mathsf{Hyb}_{3,4}(b)$ is the same as $\mathsf{Hyb}_{3,3}(b)$ since $\mathbf{z} \in S^\perp + \mathsf{cert}$ if and only if $\mathsf{cert} \in S^\perp + \mathbf{z}$.

Finally, we show that $\mathsf{Hyb}_{3,4}(b)$ outputs the error message 1 with negligible probability. This occurs precisely when the certificate check passes but the final measurement returns $I - \Pi_{\mathsf{cert}}$. Since these two projections can be simultaneously diagonalized in the Hadamard basis, we can write the probability of an error message in terms of a measurement in the Hadamard basis:

$$\Pr_{\mathsf{cert},\mathbf{z},S^\perp}[\mathbf{z} \neq \mathsf{cert} \ \wedge \ \mathbf{z} \in S^\perp + \mathsf{cert}]$$

where the probability is over the adversary outputting cert, then the challenger measuring register $\mathcal{R}_\mathcal{C}$ in the Hadamard basis to obtain $\mathbf{z}$, and finally the challenger sampling $S^\perp$. For any fixed string cert, this is a convex combination over $\mathbf{z} \neq \mathsf{cert}$ of $\Pr_{S^\perp \subset R^\perp}[\mathbf{z} \in S^\perp + \mathsf{cert}]$, so it is upper bounded by the maximum such $\mathbf{z}$. $\mathbf{z} \in S^\perp + \mathsf{cert}$ if and only if $\mathbf{z} - \mathsf{cert} \in S^\perp$, i.e. $(\mathbf{z} - \mathsf{cert}) \cdot \mathbf{s} = 0$, for all $\mathbf{s} \in S$. S (and thereby $S^\perp$) can be sampled by adjoining γ random linearly independent vectors $\mathbf{s}_i \notin R$ to $R = (R^\perp)^\perp$, which has dimension $d_S - \gamma$. For each one, the probability that $(\mathbf{z} - \mathsf{cert}) \cdot \mathbf{s}_i = 0$ is precisely $1/2$. So the overall probability of an error message is bounded above by $2^{-\gamma} = \mathsf{negl}(\lambda)$. □

□

7.2 Proof with Primal Coset Leakage

Proof. We will reduce to Theorem 7.0.3 over $\mathbb{F}_2^{d_S+\gamma}$ with the same subspace dimension d_S and same gap γ. Define $\mathcal{Z}'(S, S^\perp + \mathbf{z}, \mathbf{r}, \widetilde{b})$ as follows:

1. Sample a random change-of-basis matrix $M \in \mathbb{F}_2^{n\times n}$ over $\mathbb{F}_2^n$. We consider M to also operate on elements $x \in \mathbb{F}_2^{d_S+\gamma}$ by first appending 0s to x. Let T be the image of $\mathbb{F}_2^{d_S+\gamma}$ under M.

2. Sample $\widetilde{\mathbf{x}}' \leftarrow \mathsf{Span}(\mathbf{e}_i : i \in (d_S + \gamma, n])$ and $\mathbf{z}'_{\mathsf{re}} \leftarrow \mathbb{F}_2^n$. These will be used for rerandomizing $\mathbf{x}$ and $\mathbf{z}$.

3. Let $S_M = MS$ be the image of S under M. Define $\widetilde{\mathbf{x}} = M\widetilde{\mathbf{x}}'$.

   Let $\mathbf{z}' \in \mathbb{F}_2^n$ be the vector such that $\mathbf{z}' \cdot M\mathbf{v} = \mathbf{z} \cdot \mathbf{x}$ for all $\mathbf{v} \in \mathbb{F}_2^{d_S+\gamma}$, i.e. $\mathbf{z}' = (M^{-1})^T\mathbf{z}$. Let $\mathbf{z}_{\mathsf{re}} = \mathbf{z}' + \mathbf{z}'_{\mathsf{re}}$.

4. Sample $\mathbf{r}_2 \leftarrow \mathsf{Span}(e_i : i \in (d_S + \gamma, n])$. Set

$$\widetilde{b}' \coloneqq \widetilde{b} \oplus (\mathbf{r}_2 \cdot \widetilde{\mathbf{x}}')$$

   and set $\widetilde{\mathbf{r}} = (M^{-1})^T(\mathbf{r} + \mathbf{r}_2)$. This is the vector such that $\widetilde{\mathbf{r}} \cdot M\mathbf{v} = \mathbf{r} + \mathbf{r}_2 \cdot \mathbf{v}$ for all $\mathbf{v} \in \mathbb{F}_2^n$.

5. Output $(M,\ \widetilde{\mathbf{x}},\ \mathbf{z}'_{\mathsf{re}},\ \mathcal{Z}(S_M,\ T + \widetilde{\mathbf{x}},\ S_M^\perp + \mathbf{z}_{\mathsf{re}}))$.

We first show that $\mathcal{Z}'(S, S^\perp + \mathbf{z}, \mathbf{r}, \widetilde{b})$ is semantically-hiding in its first input and $(\dim(S_M^\perp), \dim(S_M^\perp) + \gamma, n)$-coset-hiding in its second. Consider the following hybrid distributions:

- $\mathcal{Z}''(S^\perp + \mathbf{z}, \mathbf{r}, \widetilde{b})$ does the same thing as $\mathcal{Z}'$, except it uses $\mathcal{Z}(0,\ T + \widetilde{\mathbf{x}},\ S_M^\perp + \mathbf{z}_{\mathsf{re}})$ in its output.
- $\mathcal{Z}'''(T, R^\perp + \widetilde{\mathbf{z}}, \mathbf{r}, \widetilde{b})$ behaves as $\mathcal{Z}''$ except that it replaces $\mathcal{Z}(0,\ T + \widetilde{\mathbf{x}},\ S_M^\perp + \mathbf{z}_{\mathsf{re}})$ in the output as follows. it uses $R^\perp$ to compute $R \subset S$ over $\mathbb{F}_2^{d_S+\gamma}$ and computes its embedded rerandomization $R_M \coloneqq MR \subset MS$ over $\mathbb{F}_2^n$. Finally, it uses $\mathsf{Sim}(T + \widetilde{\mathbf{x}}, R_n^\perp + \widetilde{\mathbf{z}}_{\mathsf{re}})$ in step 3, where $\widetilde{\mathbf{z}}_{\mathsf{re}} \leftarrow R_n^\perp + \mathbf{z}_{\mathsf{re}}$ and Sim is the simulator for $\mathcal{Z}$.

$\mathcal{Z}'$ and $\mathcal{Z}''$ are indistinguishable because $\mathcal{Z}$ is semantically hiding in its first input. To see that $\mathcal{Z}''$ and $\mathcal{Z}'''$ are indistinguishable, we will reduce to the $(\dim(S_M^\perp), \dim(S_M^\perp)+\gamma, n)$-coset hiding of $\mathcal{Z}$. $R^\perp \supset S^\perp$ has dimension $\gamma + \dim(S^\perp) = 2\gamma$, so $R \subset S$ is a random subspace of S with dimension $d_S + \gamma - 2\gamma = d_S - \gamma$. Therefore $R_M^\perp$ is a random superspace of $S_M^\perp$ with dimension $n - (d_S - \gamma)$. Since $S_M^\perp$ has dimension $n - d_S$, we can also write $\dim(R_M^\perp) = n - (d_S - \gamma) - \dim(S_M^\perp) + \dim(S_M^\perp) = \dim(S_M^\perp) + \gamma$. Therefore $R_M^\perp$ satisfies the requirements to invoke the coset-hiding of $\mathcal{Z}$.

Next, we give a QPT adversary for the Co-CD game using $\mathcal{Z}'$ that produces the same result as a QPT adversary for the Co-CD game using $\mathcal{Z}$.

Claim 7.2.1. *For every QPT adversary $\mathcal{A}$, there exists another QPT adversary $\mathcal{A}'$ such that*

$$\mathsf{TD}[\mathsf{Co\text{-}CD}_{\mathcal{A},\mathcal{Z}}(0), \mathsf{Co\text{-}CD}_{\mathcal{A},\mathcal{Z}}(1)] = \mathsf{TD}[\mathsf{Co\text{-}CD}_{\mathcal{A}',\mathcal{Z}'}(0), \mathsf{Co\text{-}CD}_{\mathcal{A}',\mathcal{Z}'}(1)] \tag{7.3}$$

Theorem 7.0.3 then implies that this trace distance must be negligible for all $\mathcal{A}$, completing the proof.

Proof of Claim 7.2.1. $\mathcal{A}'$ is initialized by the challenger with $|S_{\mathbf{x},\mathbf{z}}\rangle$ and

$$\mathcal{Z}(S,\ S^\perp + \mathbf{z}, \mathbf{r}, \widetilde{b}) = (M,\ \widetilde{\mathbf{x}},\ \mathbf{z}'_{re},\ \mathcal{Z}(S_M,\ T + \widetilde{\mathbf{x}},\ S'^\perp + \mathbf{z}_{re},\ \widetilde{r},\ \widetilde{b}'))$$

First, $\mathcal{A}'$ updates $|S_{x,\ z}\rangle$ to match $T + \widetilde{\mathbf{x}}$ and $S_M^\perp + \mathbf{z}_{\mathsf{re}})$. It coherently applies the unitary $U_M : |\mathbf{v}\rangle \mapsto |M\mathbf{v}\rangle$ to $|S_{\mathbf{x},\mathbf{z}}\rangle$, then applies Pauli operations $X^{\widetilde{\mathbf{x}}}$ and $Z^{\mathbf{z}'_{re}}$, resulting in the state

$$X^{\widetilde{\mathbf{x}}} Z^{\mathbf{z}'_{re}} U_M \left(X^{\mathbf{x}} Z^{\mathbf{z}} |S\rangle\right) = X^{\widetilde{\mathbf{x}}} Z^{\mathbf{z}'_{re}} \left(X^{M\mathbf{x}} Z^{\mathbf{z}'} |S_M\rangle\right) \tag{7.4}$$

$$= \pm |(S_M)_{M\mathbf{x}+\widetilde{\mathbf{x}},\ \mathbf{z}_{re}}\rangle \tag{7.5}$$

Observe that $S_M + M\mathbf{x} \subset T$ since $\mathbf{x} \in \mathbb{F}_2^{d_S+\gamma}$, so $S_M + M\mathbf{x} + \widetilde{\mathbf{x}} \subset T + \widetilde{\mathbf{x}}$. Furthermore, S_M is uniformly random within $\mathbb{F}_2^n$ since M is a random change of basis, $M\mathbf{x} + \widetilde{\mathbf{x}} = M(\mathbf{x} + \widetilde{\mathbf{x}}')$ is uniformly random within $\mathbb{F}_2^n$ since $(\mathbf{x} + \widetilde{\mathbf{x}}')$ is, and $\mathbf{z}_{\mathsf{re}}$ is uniformly random within $\mathbb{F}_2^n$ since $\mathbf{z}'_{\mathsf{re}}$ is. Also note that $\widetilde{r}$ is uniformly random within $\mathbb{F}_2^n$ because $\mathbf{r} + \mathbf{r}_2$ is.

Next, we argue that the updated state $|(S_M)_{M\mathbf{x}+\widetilde{\mathbf{x}},\, \mathbf{z}_{re}}\rangle$ together with the rerandomized values $\widetilde{\mathbf{r}}$ and $\widetilde{b}'$ encode the same bit b as the data that $\mathcal{A}'$ was initialized with. Recall that $\widetilde{\mathbf{r}} = (M^{-1})^T(\mathbf{r} + \mathbf{r}_2)$ for $\mathbf{r} \in \mathsf{Span}(\mathbf{e}_i : i \leq d_S + \gamma)$ and $\mathbf{r}_2 \in \mathsf{Span}(\mathbf{e}_i : i > d_S + \gamma)$. By the canonical rerandomization lemma (Lemma 6.2.3),

$$\widetilde{\mathbf{r}} \cdot \mathsf{Can}_{S_M}(M(x + \widetilde{x}')) = (M^{-1})^T(\mathbf{r} + \mathbf{r}_2) \cdot M\mathsf{Can}_S(x + \widetilde{x}') \tag{7.6}$$

$$= (\mathbf{r} + \mathbf{r}_2) \cdot \mathsf{Can}_S(x + \widetilde{x}') \tag{7.7}$$

$\mathbb{F}_2^{d_S+\gamma}$ and $\mathsf{Span}(\mathbf{e}_i : i \in (d_S + \gamma, n])$ are complementary with respect to $\mathbb{F}_2^n$, so we can simplify this expression using the canonical embedding lemma (Lemma 6.2.2):

$$(\mathbf{r} + \mathbf{r}_2) \cdot \mathsf{Can}_S(\mathbf{x} + \widetilde{\mathbf{x}}') = (\mathbf{r} + \mathbf{r}_2) \cdot (\mathsf{Can}_S(\mathbf{x}) \| \mathbf{0}^{n-(d_S+\gamma)} \oplus \mathbf{0}^{d_S+\gamma} \| \widetilde{\mathbf{x}}') \tag{7.8}$$

$$= (\mathbf{r} \cdot \mathsf{Can}_S(\mathbf{x})) + (\mathbf{r}_2 \cdot \widetilde{\mathbf{x}}') \tag{7.9}$$

where the last line considers $\mathsf{Can}_S(\mathbf{x})$ over $\mathbb{F}_2^{d_S+\gamma}$. Therefore

$$\widetilde{b}' \oplus (\widetilde{\mathbf{r}} \cdot \mathsf{Can}_{S_M}(M\mathbf{x} + \widetilde{\mathbf{x}}') = \widetilde{b}' \oplus (\mathbf{r} \cdot \mathsf{Can}_S(\mathbf{x})) \oplus (\mathbf{r}_2 \cdot \widetilde{\mathbf{x}}') \tag{7.10}$$

$$= \widetilde{b} \oplus (\mathbf{r} \cdot \mathsf{Can}_S(\mathbf{x})) \tag{7.11}$$

$$= b \tag{7.12}$$

Now $\mathcal{A}'$ internally runs $\mathcal{A}$ on

$$|(S_M)_{M\mathbf{x}+\widetilde{\mathbf{x}},\, \mathbf{z}_{\mathsf{re}}}\rangle \quad \text{and} \quad \mathcal{Z}(S_M,\ T + \widetilde{\mathbf{x}},\ S_M^{\perp} + \mathbf{z}_{\mathsf{re}}, \widetilde{\mathbf{r}}, \widetilde{b}')$$

Since this encodes the same bit b as the encoding that $\mathcal{A}'$ was initialized with, this is the start of $\mathsf{Co\text{-}CD}_{\mathcal{A},\mathcal{Z}}(b)$. $\mathcal{A}'$ receives from $\mathcal{A}$ a vector $\mathbf{v}$ and a state ρ_b. It computes $M^T\mathbf{v} - \mathbf{z}'_{\mathsf{re}}$ and truncates the last $n - (d_S + \gamma)$ bits to obtain $\mathbf{v}'$. Finally, it outputs $(\mathbf{v}', \rho_b)$.

Observe that if $\mathbf{v} \in S_M^{\perp} + \mathbf{z}_{\mathsf{re}}$, i.e. if it is a valid certificate for $|(S_M)_{Mx+\widetilde{x},\, \mathbf{z}_{\mathsf{re}}}\rangle$, then $\mathbf{v} = \mathbf{w} + \mathbf{z}_{\mathsf{re}}$ for some $\mathbf{w} \in S^{\perp}$. Then $M^T\mathbf{v} = M^T\mathbf{w} + (\mathbf{z} + \mathbf{z}_{\mathsf{re}}) - \mathbf{z}_{\mathsf{re}}$. Since $\mathbf{w} \in S_M^{\perp}$, it must be that $M^T\mathbf{w} \cdot \mathbf{s} = \mathbf{w} \cdot M\mathbf{s} = 0$ for all $\mathbf{s} \in S$. So $M^T\mathbf{v} \in S_n^{\perp} + \mathbf{z}$, where S_n denotes S after appending $n - (d_S + \gamma)$ bits 0 to consider it as a subspace of $\mathbb{F}_2^n$ (before rerandomizing it using M). $S_n^{\perp}$ contains precisely the elements of $S^{\perp}$ appended by every possible $(n - (d_S + \gamma))$-bit long string, since S_n is obtained by appending $n - (d_S + \gamma)$ bits 0 to S. Furthermore the last $n - (d_S + \gamma)$ bits of $\mathbf{z}$ are 0 due to applying the same embedding. Therefore discarding the last $n - (d_S + \gamma)$ bits of $M^T\mathbf{v}$ yields a vector $\mathbf{v}' \in S^{\perp} + \mathbf{z}$.

This implies that the experiment $\mathsf{Co\text{-}CD}_{\mathcal{A}',\mathcal{Z}'}(b)$ outputs ρ_b with the same distribution that $\mathsf{Co\text{-}CD}_{\mathcal{A},\mathcal{Z}}(b)$ would. In particular, Equation (7.3) holds. □

□

Chapter 8

Obfuscation with Certified Deletion

8.1 Definition

Definition 8.1.1 (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.*

- $\mathsf{diO\text{-}CD}(1^\lambda, C) \to |\widetilde{C}\rangle, \mathsf{vk}$*: 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,*

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

  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 $\{C_\lambda \in \mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$,*

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

- ***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 $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

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

- ***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 $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$,*

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

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

  - *Sample $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_\lambda$.*[1] *Sample $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(1^\lambda, C_b)$ 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 $\mathsf{diO\text{-}CD\text{-}EXP}(0, \mathcal{A}_\lambda)$ and $\mathsf{diO\text{-}CD\text{-}EXP}(1, \mathcal{A}_\lambda)$ 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* ($\mathsf{iO\text{-}CD}$). It guarantees that for any two functionally equivalent circuits C_0 and C_1, $\mathsf{iO\text{-}CD}(C_0)$ is statistically close to $\mathsf{iO\text{-}CD}(C_1)$ after they have been deleted.

Intriguingly, $\mathsf{iO\text{-}CD}$ does not imply $\mathsf{diO\text{-}CD}$ 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 diO 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 $\mathsf{diO\text{-}CD}$ 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.

8.2 Construction

We provide a construction of $\mathsf{diO\text{-}CD}$ for circuits with polynomially-many differing inputs from post-quantum iO (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 a. If the measurement is valid, then the program decrypts it to obtain two circuits, one of which is evaluated on the input a.

$\mathsf{diO\text{-}CD}$ Classical Program

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

[1] 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 diO, as it is not clearly implied by iO. However, a classical aux is sufficient for our applications.

1: **if** $\mathsf{st}_i \in T_i + \widetilde{\mathbf{x}}_i$ for every $i \in [2\ell+1]$ **then**
2: Compute $c'_i := \widetilde{c}_i \oplus (\mathsf{Can}_{S_i}(\mathbf{v}) \cdot \mathbf{r})$ for each $i \in [2\ell+1]$.
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.
4: Output $C_b(a)$.

Differing Inputs Obfuscation with Certified Deletion

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

1: Let $\ell = |C|$.
2: For each $i \in [2\ell+1]$, sample subspaces $S_i \subset T_i \subset \mathbb{F}_2^\lambda$ such that $\dim(S_i) = \lambda/2$ and $\dim(T_i) = 3\lambda/4$. Additionally for each $i \in [2\ell+1]$, sample offsets $\mathbf{x}_i, \mathbf{z}_i \leftarrow \mathbb{F}_2^\lambda$ and $\widetilde{\mathbf{x}}_i \leftarrow T_i + \mathbf{x}_i$.
3: Let $c := (0, C, 0^\ell)$. For all $i \in [2\ell+1]$, sample $\mathbf{r}_i \leftarrow \mathbb{F}_2^\lambda$ and define $\widetilde{c}_i := c_i \oplus (\mathsf{Can}_{S_i}(\mathbf{x}) \cdot \mathbf{r})$. Define $\widetilde{c} := (\widetilde{c}_1, \ldots, \widetilde{c}_{2\ell+1})$.
4: Compute the indistinguishability obfuscation $\mathsf{iO}_{\mathsf{class}}$ of the diO-CD classical program using hard-coded values $\{S_i, T_i, \widetilde{\mathbf{x}}_i, \mathbf{r}_i\}_{i\in[2\ell+1]}$ and $\widetilde{c}$.
5: For each $i \in [2\ell+1]$, compute the indistiguishability obfuscation $\mathsf{iO}_{S^\perp+\mathbf{z}} = \mathsf{iO}(P_{S^\perp+\mathbf{z}})$ of the membership checking program for $S_i^\perp + \mathbf{z}_i$.
6: Output

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

$\underline{\mathsf{Eval}(|\widetilde{C}\rangle, a)}$**:** [a]

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

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

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})}$**:** If $\mathsf{iO}_{S_i^\perp+\mathbf{z}_i}(\mathbf{v}_i) = \mathsf{Accept}$ for all $i \in [2\ell+1]$ then output Accept. Otherwise output Reject.

[a] As remarked in definition 8.1.1, one can always perform this procedure coherently and preserve the ability to run Eval on multiple inputs.

Theorem 8.2.1. *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 8.1.1).*

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.

- Hyb_0: This is the distribution $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(1^\lambda, C_0)$.
- Hyb_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 $(T_i, \widetilde{\mathbf{x}}_i)$ with $(S_i, \mathbf{x}_i)$ in $\mathsf{iO}_{\mathsf{class}}$ for each $i \in [2\ell+1]$. That is, we obfuscate the following program.

Hyb_1 Classical Program

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

1: **if** $\mathbf{v}_i \in S_i + \mathbf{x}_i$ for every $i \in [2\ell+1]$ **then**
2: Compute $c'_i := \widetilde{c}_i \oplus (\mathsf{Can}_{S_i}(\mathbf{v}_i) \cdot \mathbf{r}_i)$.
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.
4: Output $C_b(x)$.

Recall that $\mathsf{Can}(\mathbf{v}_i) = \mathsf{Can}_{S_i}(\mathbf{x}_i)$ for all $\mathbf{v}_i \in S_i + \mathbf{x}_i$. Since $\widetilde{c}_i = c_i \oplus (\mathsf{Can}_{S_i}(\mathbf{x}_i) \cdot \mathbf{r}_i)$, the program always decodes $\widetilde{c}$ to $c' = c = (0, C_0, 0^\ell)$ if it does not abort.

- Hyb_2: Replace $c = (0, C_0, 0^\ell)$ with $c = (0, C_1, 0^\ell)$ when computing $\widetilde{c}$.
- Hyb_3: For each $i \in [2\ell+1]$, replace $(S_i, \mathbf{x}_i)$ with $(T_i, \widetilde{\mathbf{x}}_i)$ in the obfuscated program P. This is the distribution $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(1^\lambda, C_1)$.

Indistinguishability between Hyb_0 and Hyb_1 and between Hyb_2 and Hyb_3 follows from subspace-hiding obfuscation (corollary 5.2.3). Indistinguishability between Hyb_1 and Hyb_2 follows from the security of diO due to the fact that C_0 and C_1 are samples from a differing inputs circuits family with a polynomial number of differing inputs. Recall that any iO is a diO for a polynomial number of differing inputs.

Certified everlasting security. Consider two differing inputs circuits $(C_0, C_1) \leftarrow \mathcal{D}_\lambda$. 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, then argue that each switch is *statistically close* conditioned on the adversary producing a successful deletion certificate.

- Hyb_0: This is the certified everlasting security game with $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(1^\lambda, C_0)$.
- Hyb_1 to Hyb_ℓ: In hybrid Hyb_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 Hyb_ℓ, the string $c = (0, C_0, C_1)$.
- $\mathsf{Hyb}_{\ell+1}$: Switch the first bit of c to 1. So, now $c = (1, C_0, C_1)$.
- $\mathsf{Hyb}_{\ell+2}$ to $\mathsf{Hyb}_{2\ell+1}$: In hybrid Hyb_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 $\mathsf{Hyb}_{2\ell+1}$, the string $c = (1, C_1, C_1)$.
- $\mathsf{Hyb}_{2\ell+2}$: Switch the first bit of c to 0. So, now $c = (0, C_1, C_1)$.
- $\mathsf{Hyb}_{2\ell+3} - \mathsf{Hyb}_{3\ell+2}$: In hybrid Hyb_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 $\mathsf{Hyb}_{3\ell+2}$, the string $c = (0, C_1, 0^\ell)$. Note that this is exactly the certified everlasting security game with $(|\widetilde{C}\rangle, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(1^\lambda, C_1)$.

The proof follows by combining the following claims.

Claim 8.2.2. $\mathsf{TD}(\mathsf{Hyb}_{j-1}, \mathsf{Hyb}_j) = \mathsf{negl}(\lambda)$ *for all* $j \in [1, \ldots, \ell]$.

Proof. We will reduce this claim to Theorem 7.0.4. To do so, we must define a distribution $\mathcal{Z}(\cdot)$ and argue that it is semantically hiding in its first input and coset-hiding in its third input. Defining $i^* := \ell + 1 + j$, we note that the output of diO-CD in Hyb_{i-1} and Hyb_i can be written as

$$|(S_{i^*})_{\mathbf{x}_{i^*},\mathbf{z}_{i^*}}\rangle \quad \text{and} \quad \mathcal{Z}_1(S_{i^*},\ T_{i^*} + \widetilde{\mathbf{x}}_{i^*},\ S_{i^*}^{\perp} + \mathbf{z}_{i^*},\ \widetilde{c}_{i^*})$$

where a sample from $\mathcal{Z}_1(S_{i^*},\ T_{i^*} + \mathbf{x}_{i^*},\ S_{i^*}^{\perp} + \mathbf{z}_{i^*},\ \widetilde{c}_{i^*})$ takes the form

$$\left(\{|(S_i)_{\mathbf{x}_i,\mathbf{z}_i}\rangle\}_{i\neq i^*}, \mathsf{iO}_{\mathsf{class}}, \left\{\mathsf{iO}\left(P_{S_i^{\perp}+\mathbf{z}_i}\right)\right\}_{i\in[2\ell+1]}\right)$$

This distribution can be sampled from by sampling from Hyb_{i-1} (equivalently, Hyb_i) using S_{i^*}, $T_{i^*} + \mathbf{x}_{i^*}$, $S_{i^*}^{\perp} + \mathbf{z}_{i^*}$, and $\widetilde{c}_{i^*}$ as hard-coded values for index i^* while sampling the other indices $i \neq i^*$ as usual.

It suffices to show that $\mathcal{Z}_1$ can be simulated using only $T_{i^*} + \widetilde{\mathbf{x}}_{i^*}$, $S_{i^*}^{\perp} + \mathbf{z}_{i^*}$, $\mathbf{r}_{i^*}$, and $\widetilde{c}_{i^*}$. This implies both required properties. We show how to simulate it via the following sequence of hybrids.

- $\mathsf{Hyb}_0(S_{i^*},\ T_{i^*} + \widetilde{x}_{i^*},\ S_{i^*}^{\perp} + \mathbf{z},\ \mathbf{r},\ \widetilde{c}_{i^*})$: This is the original distribution $\mathcal{Z}_1$ with the same inputs.
- $\mathsf{Hyb}_1(S_{i^*},\ T_{i^*} + \widetilde{x}_{i^*},\ S_{i^*}^{\perp} + \mathbf{z},\ \mathbf{r},\ \widetilde{c}_{i^*})$: This is the same as Hyb_0, except that we modify the classical program $\mathsf{iO}_{\mathsf{class}}$ 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 $(T_i, \widetilde{\mathbf{x}}_i)$ with $(S_i, \mathbf{x}_i)$ in the diO-CD classical program, for every $i \in [1, \ldots, \ell + 1]$. In other words, in Hyb_1, $\mathsf{iO}_{\mathsf{class}}$ is an obfuscation of the following program.

Hyb_1 Classical Program

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

1: **if** $\mathbf{v}_i \in S_i + \mathbf{x}_i$ for every $i \in [\ell + 1]$ **then**
2: **if** $\mathbf{v}_i \in T_i + \widetilde{x}_i$ for every $i \in [\ell + 2, 2\ell + 1]$ **then**
3: Compute $c'_i := \widetilde{c}_i \oplus (\mathsf{Can}_{S_i}(\mathbf{v}_i) \cdot \mathbf{r}_i)$.
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.
5: Output $C_b(x)$.

  Note that if $\mathbf{v}_i \in S_i + \mathbf{x}_i$, then $\mathsf{Can}_{S_i}(\mathbf{v}_i) = \mathsf{Can}_{S_i}(\mathbf{x}_i)$. This is always the case for $i \in [\ell + 1]$ if the program does not abort. Since $\widetilde{c}_i = c_i \oplus (\mathsf{Can}_{S_i}(\mathbf{x}_i) \cdot \mathbf{r}_i)$, the program always decodes the first $\ell + 1$ bits of $\widetilde{c}$ to $(0, C_0)$ if it does not abort.

- $\mathsf{Hyb}_2(T_{i^*} + \widetilde{x}_{i^*},\ S_{i^*}^{\perp} + \mathbf{z},\ \mathbf{r},\ \widetilde{c}_{i^*})$: This is the same as Hyb_1, except that we replace S_{i^*} with $0^{|S_{i^*}|}$ in the classical obfuscated program iO_{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.
- $\mathsf{Hyb}_3(T_{i^*} + \widetilde{x}_{i^*},\ R_{i^*}^{\perp} + \widetilde{\mathbf{z}},\ \mathbf{r},\ \widetilde{c}_{i^*})$: This is the same as Hyb_2, except that we will remove the dependence of $\mathsf{iO}\left(P_{S_{i^*}^{\perp}+\mathbf{z}_{i^*}}\right)$ on $S_{i^*}^{\perp} + \mathbf{z}_{i^*}$. Sample R_{i^*} as a uniformly random superspace of $S_{i^*}^{\perp}$ of dimension $3\lambda/4$ and set $\widetilde{\mathbf{z}}_{i^*} = \mathsf{Can}_{R_{i^*}^{\perp}}(\mathbf{z}_{i^*})$. Use $\mathsf{iO}\left(P_{R_{i^*}^{\perp}+\widetilde{\mathbf{z}}_{i^*}}\right)$ in place of $\mathsf{iO}\left(P_{S_{i^*}^{\perp}+\mathbf{z}_{i^*}}\right)$. Note

that if we consider $T_{i^*} + \widetilde{x}_{i^*}$, $\mathbf{r}_{i^*}$, and $\widetilde{c}_{i^*}$ to be fixed, then this distribution depends only on $R^{\perp}_{i^*} + \widetilde{\mathbf{z}}_{i^*}$, so it can be considered as a coset-hiding simulator for definition 7.0.1.

The indistinguishability of Hyb_0 and Hyb_1 follows by repeated application of subspace-hiding obfuscation (corollary 5.2.3) for each $i \in [1, \ldots, \ell + 1]$. Next, the indistinguishability of Hyb_1 and Hyb_2 follows from the security of iO, 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 Hyb_2 and Hyb_3 follows again from corollary 5.2.3.

□

Claim 8.2.3. $\mathsf{TD}(\mathsf{Hyb}_\ell, \mathsf{Hyb}_{\ell+1}) = \mathsf{negl}(\lambda)$.

Proof. We will again reduce this claim to theorem 7.0.4. Note that the output of $\mathsf{diO\text{-}CD}$ in Hyb_ℓ and $\mathsf{Hyb}_{\ell+1}$ can be written as

$$|(S_1)_{\mathbf{x}_1,\mathbf{z}_1}\rangle \quad \text{and} \quad \mathcal{Z}_1(S_1,\ T_1 + \widetilde{\mathbf{x}}_1,\ S_1^{\perp} + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$$

where a sample from $\mathcal{Z}_1(S_1,\ T_1 + \widetilde{\mathbf{x}}_1,\ S_1^{\perp} + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$ takes the form

$$\left(\{|(S_i)_{\mathbf{x}_i,\mathbf{z}_i}\rangle\}_{i\neq 1},\ \mathsf{iO}_{\mathsf{class}},\ \left\{\mathsf{iO}\left(P_{S_i^{\perp}+\mathbf{z}_i}\right)\right\}_{i\in[2\ell+1]}\right)$$

This distribution can be sampled from by sampling from Hyb_ℓ (equivalently, $\mathsf{Hyb}_{\ell+1}$) using S_1, $T_1 + \mathbf{x}_1$, $S_1^{\perp} + \mathbf{z}_1$, $\mathbf{r}_1$, and $\widetilde{c}_1$ as hard-coded values for index 1 while sampling the other indices $i \neq 1$ as usual. In Hyb_ℓ we have $\widetilde{c}_1 = 0 \oplus (\mathsf{Can}_{S_1}(\mathbf{x}_1) \cdot \mathbf{r}_1)$, while in $\mathsf{Hyb}_{\ell+1}$ we have $\widetilde{c}_1 = 1 \oplus (\mathsf{Can}_{S_1}(\mathbf{x}_1) \cdot \mathbf{r}_1)$.

It suffices to show that $\mathcal{Z}_1$ can be simulated using only $T_1 + \widetilde{\mathbf{x}}_1$, $S_1^{\perp} + \mathbf{z}_1$, $\mathbf{r}_1$, and $\widetilde{c}_1$. This implies both required properties. We show how to simulate it via the following sequence of hybrids.

- $\mathsf{Hyb}_0(S_1,\ T_1 + \widetilde{\mathbf{x}}_1,\ S_1^{\perp} + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$: This is $\mathcal{Z}_1$ with the same inputs.
- $\mathsf{Hyb}_1(S_1,\ T_1 + \widetilde{\mathbf{x}}_1,\ S_1^{\perp} + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$: This is the same as Hyb_0, except that we modify the classical obfuscated program $\mathsf{iO}_{\mathsf{class}}$ 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 $T_i + \widetilde{\mathbf{x}}_i$ with $S_i + \mathbf{x}_i$ in the $\mathsf{diO\text{-}CD}$ classical program for every $i \in [2, \ldots, 2\ell + 1]$.[2]

  Note that if $\mathbf{v}_i \in S_i + \mathbf{x}_i$, then $\mathsf{Can}_{S_i}(\mathbf{v}_i) = \mathsf{Can}_{S_i}(\mathbf{x}_i)$. This is always the case for $i > 1$ if the program does not abort. Since $\tilde{c}_i = c_i \oplus (\mathsf{Can}_{S_i}(\mathbf{x}_i) \cdot \mathbf{r}_i)$, the program always decodes the last $2\ell+$ bits of $\widetilde{c}$ to (C_0, C_1) if it does not abort.
- $\mathsf{Hyb}_2(S_1,\ T_1 + \widetilde{\mathbf{x}}_1,\ S_1^{\perp} + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$: This is the same as Hyb_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, $\mathsf{iO}_{\mathsf{class}}$ is an obfuscation of the following program:

Hyb_2 Classical Program

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

[2] This is similar to $\mathsf{iO}_{\mathsf{class}}$ for Hyb_1 in the proof of the prior Claim 8.2.2.

1: **if** $\mathbf{v}_1 \in T_1 + \widetilde{\mathbf{x}}_1$ **then**
2: **if** $\mathbf{v}_i \in S_i + \mathbf{x}_i$ for every $i \in [1, \ldots, 2\ell+1]$ **then**
3: Compute $b := \widetilde{c}_1 \oplus (\mathsf{Can}_{S_1}(\mathbf{v}_1) \cdot \mathbf{r}_1)$.
4: Output $\widetilde{C}_b(a)$.

- $\mathsf{Hyb}_3(S_1,\ T_1 + \widetilde{\mathbf{x}}_1,\ S_1^\perp + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$: Same as Hyb_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, $\mathsf{iO}_{\mathsf{class}}$ is an obfuscation of the following program:

Hyb_3 Classical Program

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

1: **if** $\mathbf{v}_1 \in T_1 + \widetilde{\mathbf{x}}_1$ **then**
2: **if** $\mathbf{v}_i \in S_i + \mathbf{x}_i$ for every $i \in [1, \ldots, 2\ell+1]$ **then**
3: Compute $b := \widetilde{c}_1 \oplus (\mathsf{Can}_{S_1}(\mathbf{v}_1) \cdot \mathbf{r}_1)$.
4: Output $\widetilde{C}_b(a)$.

- $\mathsf{Hyb}_4(T_1 + \widetilde{\mathbf{x}}_1,\ S_1^\perp + \mathbf{z}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$: This is the same as Hyb_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, $\mathsf{iO}_{\mathsf{class}}$ is an obfuscation of the following program:

Hyb_4 Classical Program

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

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

- $\mathsf{Hyb}_5(T_1 + \widetilde{\mathbf{x}}_1,\ R_1^\perp + \widetilde{vz}_1,\ \mathbf{r}_1,\ \widetilde{c}_1)$: This is the same as Hyb_4, except that we will remove the dependence of $\mathsf{iO}\left(P_{S_1^\perp + \mathbf{z}_1}\right)$ on $S_1^\perp + \mathbf{z}_1$. Sample $R_1^\perp$ as a uniformly random superspace of $S_1^\perp$ of dimension $3\lambda/4$ and set $\widetilde{\mathbf{z}}_1 := \mathsf{Can}_{R^\perp}(\mathbf{z}_1)$. Use $\mathsf{iO}\left(P_{R_1^\perp + \widetilde{\mathbf{z}}_1}\right)$ in place of $\mathsf{iO}\left(P_{S_1^\perp + \mathbf{z}_1}\right)$. Note that if we consider the other parameters to be fixed, then this distribution depends only on $R_1^\perp + \widetilde{\mathbf{z}}_1$, so it can be consider as a coset-hiding simulator for Definition 7.0.1.

Now, the indistinguishability of Hyb_0 and Hyb_1 follows by repeated application of corollary 5.2.3 for each $i \in [2, \ldots, 2\ell+1]$.

Next, the indistinguishability of Hyb_1 and Hyb_2 follows from the security of iO, since these programs are functionally equivalent. Indeed, note that in Hyb_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 Hyb_1 and Hyb_2, the program outputs $C_b(x)$ if

it does not abort.

The indistinguishability of Hyb_2 and Hyb_3 follows from the security of diO, which follows from the security of iO. Since C_0 and C_1 differ on a polynomial number of hard-to-find inputs, $\mathsf{iO}(C_0) \approx \mathsf{iO}(C_1)$. In more detail, assuming Hyb_2 and Hyb_3 are distinguishable, we can distinguish $\mathsf{iO}(C_0)$ from $\mathsf{iO}(C_1)$ as follows. The reduction is given (C_0, C_1, aux) sampled by the differing inputs circuits sampler, as well as $\mathsf{iO}(C_b)$ for a random b. It classically samples the cosets, then constructs the obfuscated program specified by Hyb_2/Hyb_3 using $\mathsf{iO}(C_b)$. It runs the distinguisher for Hyb_2/Hyb_3 and outputs the result.

The indistinguishability of Hyb_3 and Hyb_4 follows from the security of iO, since these programs are functionally equivalent. Indeed, note that in Hyb_3, the program will always evaluate $\mathsf{iO}(C_1)$ if it does not abort, which is functionally equivalent to C_1. Finally, the indistinguishability of Hyb_4 and Hyb_5 follows again from corollary 5.2.3.

□

Claim 8.2.4. $\mathsf{TD}(\mathsf{Hyb}_{i-1}, \mathsf{Hyb}_i) = \mathsf{negl}(\lambda)$ *for all* $i \in [\ell + 2, \ldots, 2\ell + 1]$.

Proof. This follows from essentially an identical proof as claim 8.2.2. □

Claim 8.2.5. $\mathsf{TD}(\mathsf{Hyb}_{2\ell+1}, \mathsf{Hyb}_{2\ell+2}) = \mathsf{negl}(\lambda)$.

Proof. This follows from a similar proof to claim 8.2.3. The only difference is that we can transition directly from sub-hybrid Hyb_1 to Hyb_4 using the security of iO. Note that in Hyb_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 Hyb_4, the classical program also always evaluates C_1 if it does not abort, and the aborting conditions are the same. □

Claim 8.2.6. $\mathsf{TD}(\mathsf{Hyb}_{i-1}, \mathsf{Hyb}_i) = \mathsf{negl}(\lambda)$ *for all* $i \in [2\ell + 3, \ldots, 3\ell + 2]$.

Proof. This follows from essentially an identical proof as claim 8.2.2. □

□

8.3 Variant: Nesting

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 iO 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.[3] 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 $\mathsf{diO\text{-}CD}$.

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

Definition 8.3.1 (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 $C_f(x, y, C_{\mathcal{C}}(y))$. It is associated with a distribution $\mathcal{D}'_{\mathcal{C}}$. 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 $(C_f \circ C_0, C_f \circ C_1, \mathsf{aux}) \leftarrow \mathcal{D}'$ if it is a differing input for (C_0, C_1). Note that two circuits $C_f \circ C_0$ and $C_f \circ C_1$ 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 $\mathsf{diO\text{-}CD}$ security game. We say a $\mathsf{diO\text{-}CD}$ 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.

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

Corollary 8.3.2. *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.*

8.4 Variant: 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 $\mathsf{diO\text{-}CD}$ construction does not allow for this. Since the functionality of an obfuscated program $\widetilde{C} = (|\psi\rangle, \mathsf{iO}_{\mathsf{class}})$ is determined by $|\psi\rangle$, whether or not $\widetilde{C}$ has the correct functionality is not a QMA statement.

Tokenized $\mathsf{diO\text{-}CD}$ To remedy this, we introduce a new $\mathsf{diO\text{-}CD}$ property which we call "tokenization" and give an alternative scheme which satisfies it. A tokenized $\mathsf{diO\text{-}CD}$ 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 $\mathsf{diO\text{-}CD}$ 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 8.4.1 (Tokenized $\mathsf{diO\text{-}CD}$)**.** *A (nested) differing inputs obfuscator with certified deletion* $(\mathsf{Obf}, \mathsf{Eval}, \mathsf{Del}, \mathsf{Verify})$ *is* tokenized *if it satisfies the following properties:*

1. $\mathsf{Obf}(C)$ *outputs a quantum token* $|\psi\rangle = \sum_{t \in \{0,1\}^{\mathsf{poly}(\lambda)}} \alpha_t |t\rangle$ *and a classical program* $\widetilde{C}$.

2. $\mathsf{Eval}((|\psi\rangle, \widetilde{C}), x) = \widetilde{C}(|\psi\rangle, x) = \sum_{t \in \{0,1\}^{\mathsf{poly}(\lambda)}} \alpha_t |\widetilde{C}(t, x)\rangle$.

3. *The probability of generating an obfuscation* $(|\psi\rangle, \widetilde{C}) \leftarrow \mathsf{Obf}(C)$ *such that for all* $t \in \{0,1\}^{\mathsf{poly}(\lambda)}$*, either*

   (a) for all x*,* $\widetilde{C}(t, x) = C(x)$

   (b) or for all x*,* $\widetilde{C}(t, x) = \bot$

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

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 8.4.2 (Computational certified deletion for $\mathsf{diO\text{-}CD}$)**.** *Let* $\mathcal{D}_{\mathcal{C}}$ *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* $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{D}_{\mathcal{C}}$*. It computes the set of differing inputs* $Y^* = \{y^* : C_0(y^*) \neq C_1(y^*)\}$ *and the obfuscation* $(\widetilde{C_b}, \mathsf{vk}) \leftarrow \mathsf{diO\text{-}CD}(C_b, 1^\lambda)$*, 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 $\mathsf{diO\text{-}CD}$ *scheme has computational certified deletion security for* $\mathcal{C}$ *if the adversary's advantage in winning this game is* $\mathsf{negl}(\lambda)$.

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

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

- Let $\ell = |C|$. For each $i \in [2\ell + 1]$, sample subspaces $S_i \subset \mathbb{F}_2^n$ with $\dim(S_i) = \lambda/2$, then sample $\mathbf{x}_i, \mathbf{z}_i, \mathbf{r}_i \leftarrow \mathbb{F}_2^\lambda$.
- Let $c := (0, C, 0^\ell)$, and for all $i \in [2\ell + 1]$, define $\widetilde{c}_i := c_i \oplus (\mathsf{Can}_{S_i}(\mathbf{x}_i) \cdot \mathbf{r}_i)$. Define $\widetilde{c} := (\widetilde{c}_1, \ldots, \widetilde{c}_{2\ell+1})$.
- Compute the indistinguishability obfuscation $\mathsf{iO}_{\mathsf{class}}$ of the $\mathsf{diO\text{-}CD}$ classical program using hard-coded values $\{S_i, S_i, \mathbf{x}_i, \mathbf{r}_i\}_{i \in [2\ell+1]}$ and $\widetilde{c}$.

- Output

$$|\widetilde{C}\rangle \coloneqq \left(\{|(S_i)_{\mathbf{x}_i,\mathbf{z}_i}\rangle\}_{i\in[2\ell+1]},\ \mathsf{iO}_{\mathsf{class}}\right) \quad \text{and} \quad \mathsf{vk} \coloneqq \left\{\mathsf{iO}\left(P_{S_i^\perp+\mathbf{z}_i}\right)\right\}_{i\in[2\ell+1]}.$$

Deletion and verification are done as in the original scheme. For convenience, we recall the diO-CD classical program here, updated for the chosen parameters.

diO-CD Classical Program

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

1: **if** $\mathbf{v}_i \in S_i + \mathbf{x}_i$ for every $i \in [2\ell+1]$ **then**
2: Compute $c'_i \coloneqq \widetilde{c}_i \oplus (\mathsf{Can}_{S_i}(\mathbf{x}_i) \cdot \mathbf{r}_i)$.
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.
4: Output $C_b(a)$.

Corollary 8.4.3 (Tokenized diO-CD)**.** *Assuming post-quantum indistinguishability obfuscation and injective one-way functions, there exists* diO-CD with provable correctness *for polynomially many differing inputs.*

Proof. Observe that if $\mathbf{v}_i \in S_i + \mathbf{x}_i$, 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 a, the above construction is indeed tokenized. This establishes provable correctness.

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

- Hyb_0: The outcome bit from the computational certified deletion game for the tokenized diO-CD construction.
- Hyb_1: This is the same as Hyb_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 $3\lambda/4$. Let $\widetilde{\mathbf{x}}_i \coloneqq \mathsf{Can}_{T_i}(\mathbf{x}_i)$ Instead of obfuscating the diO-CD classical program with $\{S_i, S_i, \mathbf{x}_i, \mathbf{r}\}$ hard-coded, use $\{S_i, T_i, \widetilde{\mathbf{x}}_i, \mathbf{r}\}$. This is our original diO-CD scheme.

Indistinguishability of Hyb_0 and Hyb_1 follows from the repeated application of subspace-hiding obfuscation (corollary 5.2.3). Observe that any QPT adversary's winning advantage in Hyb_1 is negligible. Otherwise, an adversary could violate the (information-theoretic) certified deletion security of the original diO-CD scheme (theorem 8.2.1) 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 Hyb_0 and Hyb_1 are indistinguishable, this also holds in Hyb_0. □

Chapter 9

Applications

In this section, we discuss several applications of our techniques:

- **Section 9.1: Encryption with Publicly Verifiable Certified Deletion.** Encryption with certified deletion guarantees that once an adversary generates a valid certificate for the ciphertext, the plaintext is information-theoretically lost. We show how to publish the verification key while still maintaining this guarantee, assuming indistinguishability obfuscation and injective one-way functions.
- **Section 9.2: Functional Encryption with (Publicly Verifiable) Certified Deletion for Ciphertexts.** Functional encryption allows someone who possesses a key sk_f for a function f and a ciphertext ct_m for a message m to compute $f(m)$, but nothing more. Certified deletion for ciphertexts adds the ability to verifiably destroy the ciphertext ct_m. Afterwards, the remaining information about m is information-theoretically lost; nothing more can be learned about it even if additional keys sk_g are acquired. We show how to construct this from sub-exponentially secure obfuscation with certified deletion.
- **Section 9.3: Functional Encryption with (Publicly Verifiable) Key Revocation.** In contrast, key revocation adds the ability to delete the decryption key sk_f. After deleting sk_f, even if an adversary learns a new ciphertext $\mathsf{ct}_{m'}$, they will not be able to learn $f(m')$.[1] We show how to construct this from obfuscation with certified deletion and injective one-way functions.
- **Section 9.4: Strong Secure Software Leasing.** Consider a leasor who leases out a piece of software to a leasee. Secure software leasing guarantees that if the leasee returns an intact (as determined by the leasor's verification) copy of the software, then they cannot also have a fully functional copy of the software locally that can be evaluated using the *honest* evaluation procedure. Strong secure software leasing ensures that the copy cannot be evaluated even using arbitrary evaluation procedures. We show that this is immediately implied by obfuscation with certified deletion.

9.1 Encryption with Publicly Verifiable Certified Deletion

Standard certified deletion requires that the verification key vk is hidden from the adversary. It would be useful if vk could be safely published, allowing anyone to verify certificates of deletion. Combining our coset certified deletion theorem with constructions from [BK23] yields a variety of primitives with *publicly verifiable* certified deletion.

[1] This is the only result in this section which is not information-theoretically secure after deletion. This weakness is inherent, since an unbounded adversary could anyway break $\mathsf{ct}_{m'}$ to recover m' without ever having received sk_f.

Corollary 9.1.1. *Assuming indistinguishability obfuscation and post-quantum $X \in \{$public-key, attribute-based, fully-homomorphic, time-release, witness$\}$ encryption, there exists X encryption with publicly verifiable certified deletion.*

We briefly sketch the construction for public-key encryption. The other constructions are analogous. Key generation simply generates a key pair $(\mathsf{sk}, \mathsf{pk})$ from the underlying (post-quantum) PKE. To encrypt a message bit[2] $m \in \{0,1\}$ using public key pk, sample an $\lambda/2$-dimensional subspace $S \subset \mathbb{F}_2^\lambda$ and offsets $x, z \leftarrow \mathbb{F}_2^\lambda$, then output

$$\mathsf{ct} = |S_{\mathbf{x},\mathbf{z}}\rangle\,,\ r,\ (r \cdot \mathsf{Can}_S(\mathbf{x})) \oplus m,\ \mathsf{PKE.Enc}(\mathsf{pk}, S) \tag{9.1}$$

$$\mathsf{vk} = \mathsf{iO}(P_{S^\perp + \mathbf{z}}) \tag{9.2}$$

where $P_{S^\perp + z}$ decides membership in $S^\perp + \mathbf{z}$. Decryption can be done using S, which can be recovered from $\mathsf{PKE.Enc}(\mathsf{pk}, S)$ using sk. To delete a ciphertext, measure $|S_{\mathbf{x},\mathbf{z}}\rangle$ in the Hadamard basis to get a vector $\mathbf{v} \in S^\perp + \mathbf{z}$. Anyone holding vk can check that $\mathbf{v} \in S^\perp + \mathbf{z}$ to validate the certificate $\mathbf{v}$. Deletion security is immediate from our coset deletion theorem using the observation that $(\mathsf{PKE.Enc}(\mathsf{pk}, S),\ \mathsf{iO}(P_{S^\perp + z}))$ is subspace-hiding.

9.2 Functional Encryption with Certified Deletion for Ciphertexts

Next, we construct a new kind of encryption where the ciphertexts can be verifiably deleted: functional encryption. In functional encryption, a central party can distribute secret keys sk_f which allow the holder of sk_f to learn $f(x)$ from $\mathsf{Enc}(x)$, but no more. Ciphertext certified deletion additionally requires that the ciphertexts $\mathsf{Enc}(x)$ can be verifiably deleted. Afterwards, the adversary cannot learn any more information about x even if it receives a new key sk_g or breaks the computational problem underlying the functional encryption scheme.

In contrast to other encryption schemes, functional encryption reveals some information about x even before $\mathsf{Enc}(x)$ is deleted. This closely matches how obfuscation with certified deletion reveals some information about the program before it is deleted.[3]

In the following we sketch how to construct FE with ciphertext certified deletion, albeit at the cost of assuming subexponential hardness of the underlying building blocks. We leave open the problem of constructing FE with ciphertext certified deletion from polynomial assumptions.

9.2.1 Definition

Syntax. FE with ciphertext certified deletion augments the syntax of a standard FE scheme with two new algorithms Del and Verify. Additionally, Enc also outputs a verification key vk and all algorithms may be QPT.

- $\mathsf{Del}(|\mathsf{ct}\rangle)$ takes as input a ciphertext and outputs a certificate.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert})$ takes as input a verification key vk and a certificate cert, then outputs Accept or Reject.

The scheme needs to satisfy an additional correctness property, that Verify will always accept a certificate generated honestly from the ciphertext:

$$\Pr[\mathsf{Verify}(\mathsf{Del}(c), \mathsf{vk}) = \mathsf{Acc} :\ (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\ (c, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m)] = 1.$$

[2] This can be generically lifted to longer messages via a standard hybrid argument.

[3] Even in the classical setting, functional encryption and obfuscation are closely related [BV15, AJ15, GMM17].

Security. Security can be defined using a variant of the FE security game (Definition 5.3.3) where the query restrictions on secret keys sk_f are completely lifted after the adversary deletes the challenge ciphertext ct. More explicitly: during the entire game, the adversary may query for secret keys sk_f. At any point, it may query for a challenge ciphertext $\mathsf{ct}^* = \mathsf{Enc}(m_b)$ by submitting two messages m_0 and m_1 to the challenger. At any point, it may then submit a certificate. The only constraint on the adversary's queries is that $f(m_0) = f(m_1)$ for all f queried *before* the challenger receives a valid certificate. Afterwards, the adversary is free to query the key generation oracle on *any functions*.

We also consider a more stringent security definition where the adversary becomes *computationally unbounded* after submitting a valid certificate. In this case, it could directly compute new secret keys sk_f itself.

Definition 9.2.1 (FE Certified Everlasting Security for Ciphertexts)**.** *A functional encryption with certified deletion for ciphertexts has* certified everlasting security *if for every QPT adversary $\mathcal{A}$,*

$$\mathsf{TD}[\mathsf{FE\text{-}CEV}_{\mathcal{A}}(0), \mathsf{FE\text{-}CEV}_{\mathcal{A}}(1)] = \mathsf{negl}(\lambda)$$

where $\mathsf{FE\text{-}CEV}_{\mathcal{A}}(b)$ is defined as follows:

1. ***Setup:*** *The challenger generates a key pair $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends pk to $\mathcal{A}$.*
2. ***Query Phase 1:*** *The adversary may query functions $f \in \mathcal{F}_\lambda$ to the challenger. The challenger replies with $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$.*
3. ***Challenge:*** *$\mathcal{A}$ sends two messages (m_0, m_1) to the challenger, subject to the constraint that $f(m_0) = f(m_1)$ for all previously queried f. The challenger sends back ct from $(\mathsf{ct}, \mathsf{vk}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$.*
4. ***Query Phase 2:*** *The adversary may submit additional key queries for functions f, subject to the constraint that $f(m_0) = f(m_1)$.*
5. ***Deletion and Output.*** *The adversary outputs a certificate cert and a state ρ. If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cert}) = \mathsf{Accept}$, output ρ. Otherwise, output $\bot$.*

We say the functional encryption scheme has selective *security 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 is publicly verifiable *if this holds even when $\mathcal{A}$ also receives vk during the challenge step.*

9.2.2 Construction

The ingredient of the construction are a subexponentially-secure $\mathsf{diO\text{-}CD}$ scheme and a unique signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Verify})$. The public parameters of the scheme simply consist of a verification key vk and a signing key sk sampled uniformly. A functional key for f is computed as

$$\mathsf{sk}_f \leftarrow \mathsf{Sig}(\mathsf{sk}, f).$$

To encrypt a message m, define Π to be the program that takes as input a function f and a signature σ, checks the validity of the signature $1 = \mathsf{Verify}(\mathsf{vk}, f, \sigma)$ and if this check passes, outputs $f(m)$. Otherwise it outputs $\bot$. The ciphertext is then defined to be the $\mathsf{diO\text{-}CD}$ of Π. Decryption works canonically, and correctness follows by the correctness of the underlying building blocks.

Theorem 9.2.2. *Assuming sub-exponentially*[4] *secure differing inputs obfuscation with certified deletion and unique signatures, there exists functional encryption with selective ciphertext certified everlasting security. Furthermore, the certificates are publicly verifiable.*

Sketch. Next, we provide a sketch of the security analysis. The proof proceeds by defining a series of hybrids, iterating over all possible functions f. Starting from an encryption of m_0, each hybrid gradually modifies the obfuscated circuit to output $f(m_1)$ on input a valid pair (f, σ). Note that this circuit can be defined in such a way that its size does not blow up exponentially (e.g., by iterating over all f in lexicographical order). To argue indistinguishability, we consider two cases:

- If $f(m_0) = f(m_1)$, then indistinguishability follows from the sub-exponential security of $\mathsf{diO\text{-}CD}$, since the two circuits are functionally equivalent, i.e. there are zero differing inputs.
- If $f(m_0) \neq f(m_1)$, then observe that the two circuits differ in exactly one input, which is (f, σ), where σ is the unique valid signature on f. By the sub-exponential unforgeability of the signature scheme, it follows that the two circuits are a valid differing-input pair. Thus indistinguishability follows by the sub-exponential security of the $\mathsf{diO\text{-}CD}$, since the adversary can only query the signature (i.e., the differing-input) only after producing a valid deletion certificate.

In the final hybrid, the circuit is functionally equivalent to a valid encryption of m_1 and therefore we can conclude the proof by another invocation of the sub-exponential security of $\mathsf{diO\text{-}CD}$. □

Unique Signatures. We conclude by recalling the unique signatures can be instantiated using standard iO and one-way function, using the construction of Sahai and Waters [SW14]. In their scheme, a signatures on a message m is the output of a pseudorandom function (PRF) on m, whereas the verification key consists of an obfuscated circuit that checkes whether the PRF was correctly computing (using the hardwired key). Since the PRF is in particular a function, it follows that signatures are unique. For more details, we refer the reader to [SW14].

9.3 Functional Encryption with Key Revocation

We can also consider a dual notion of certified deletion for FE, where the *secret key* sk_f can be verifiably deleted. After sk_f is deleted, the party is no longer able to learn $f(x)$ from new ciphertexts $\mathsf{Enc}(x)$ (assuming they do not have a second sk'_f for the same function which they did not delete).

We show that we can construct FE with key revocation using differing inputs obfuscation with certified deletion. We note that our construction can also be used to generically construct key revocation for any PKE.

9.3.1 Definition

Definition 9.3.1 (Functional Encryption with Key Revocation)**.** *A functional encryption scheme with key revocation is associated with a class of functions $\mathcal{F}(\lambda)$ and a message space $\mathcal{M}_\lambda$. It consists of the following QPT algorithms:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter λ, then outputs a public key* pk *and a master secret key* msk.

[4]This refers to the final trace distance.

- $\mathsf{KeyGen}(\mathsf{msk}, f)$ *takes as input the master secret key* msk *and the description of a function* $f \in \mathsf{F}$*, then outputs a secret key* sk_f *and a verification key* vk*.*
- $\mathsf{Enc}(\mathsf{pk}, m)$ *takes as input the public key* pk *and a message* m*, then outputs a ciphertext* c*.*
- $\mathsf{Dec}(\mathsf{sk}_f, c)$ *takes as input a secret key* sk_f *and a ciphertext encrypting a message* $m \in \mathcal{M}_\lambda$*, then outputs* $f(m)$*.*
- $\mathsf{Del}(\mathsf{sk}_f)$ *takes as input a decryption key, then outputs a certificate of deletion* cert*.*
- $\mathsf{Verify}(\mathsf{cert}, \mathsf{vk})$ *takes as input a certificate of deletion* cert *and a verification key* vk*, then outputs* Accept *or* Reject*.*

Definition 9.3.2 (Correctness for FE with Key Revocation)**.** *A functional encryption scheme with key revocation is correct if it satisfies the following two properties:*

- ***Decryption Correctness*** *For all functions* $f \in \mathsf{F}$ *and messages* $m \in \mathsf{M}$*,*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{pk}, m)) \neq f(m) : \begin{array}{c}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{sk}_f, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)\end{array}\right] = \mathsf{negl}(\lambda)$$

- ***Deletion Correctness***

$$\Pr[\mathsf{Verify}(\mathsf{Del}(\mathsf{sk}_f), \mathsf{vk}) = \mathsf{Acc} : (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_f, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)] = 1$$

In the key revocation game, the adversary receives some number of secret keys for functions of its choice. It deletes some of them, then queries the challenger on messages m_0 and m_1 such that $f(m_0) = f(m_1)$ for every secret key sk_f which it did *not* delete. There are no restrictions with respect to the functions computed by deleted secret keys. After receiving a ciphertext for one of the messages, the adversary then is allowed to receive more secret keys sk_f subject to the constraint that $f(m_0) = f(m_1)$. They then attempt to guess which message was encrypted.

Definition 9.3.3 (Key Revocation for FE)**.** *A functional encryption scheme, augmented as above, has secret key certified deletion if the advantage of every QPT adversary in the following game is* $\mathsf{negl}(\lambda)$*:*

1. *The challenger samples* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to the adversary.*
2. *The following query phase is repeated a polynomial number of times:*
   (a) *The adversary adaptively submits a query* $f_i \in \mathcal{F}(\lambda)$*.*
   (b) *The challenger samples* $(\mathsf{sk}_{f_i}, \mathsf{vk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$ *and sends* sk_{f_i} *to the adversary.*
3. *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* $\mathsf{cert}_i = \bot$*.*
4. *The challenger checks the deletion proofs. If* $f_i(m_0) = f_i(m_1)$ *for every* f_i *such that* $\mathsf{Verify}(\mathsf{cert}_i, \mathsf{vk}_i) = \mathsf{Rej}$*, then sample a random bit* b *and send* $\mathsf{Enc}(\mathsf{pk}, m_b)$ *to the adversary.*
5. *The following query phase is repeated a polynomial number of times:*
   (a) *The adversary adaptively submits a query* $f_i \in \mathcal{F}(\lambda)$*.*

(b) If $f_i(m_0) = f_i(m_1)$, the challenger samples $(\mathsf{sk}_{f_i}, \mathsf{vk}_i)$ and sends sk_{f_i} to the adversary.

6. *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* $(\mathsf{pk}, \mathsf{sk})$.

Note that key revocation security subsumes standard indistinguishability security for FE (Definition 5.3.3).

Remark 9.3.4. *Definition 9.3.3 only requires computational security after deletion, unlike many of our other results. Unfortunately, this is inherent. Even if the adversary never received a secret key sk_f, the encryption of any message is still only computationally secure for any FE scheme.*

9.3.2 Construction

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 diO-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

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

1: Sample two classical PKE key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{Gen}_{\mathsf{PKE}}(1^\lambda)$.
2: Set $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{NIZK}}(1^\lambda)$.

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 $\mathsf{msk} = \mathsf{sk}_1$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, f)}$ Output a diO-CD program for the FE Secret Key Functionality using the parameters $(f, \mathsf{pk}, \mathsf{sk}_1)$.

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$

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 $(\pi, c_1, c_{2,1}, c_{2,2})$.

$\underline{\mathsf{Dec}(\mathsf{sk}_f, c)}$

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 $\widetilde{C}_{\mathsf{sk}_f}(|t\rangle, \pi, c_1, c_{2,1}, c_{2,2})$.

$\underline{\mathsf{Del}(\mathsf{sk}_f)}$

1: Parse $\mathsf{sk}_f = (|t\rangle, \widetilde{C}_{\mathsf{sk}_f})$.
2: Compute and output the diO-CD deletion proof $\mathsf{cert} \leftarrow \mathsf{Del}_{\mathsf{diO\text{-}CD}}(|t\rangle)$.

$\underline{\mathsf{Verify}(\mathsf{cert}, \mathsf{vk})}$ Output $\mathsf{Verify}_{\mathsf{diO\text{-}CD}}(\mathsf{cert}, \mathsf{vk})$.

Theorem 9.3.5 (Selective FE with Key Revocation)**.** *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. The assumptions in this theorem can be reduced to post-quantum iO and injective one-way functions. We previously constructed nested diO-CD from iO and injective one-way functions (Corollary 8.3.2). Public-key encryption can also be constructed from iO and one-way functions [GGH+13, GGSW13]. Their security proofs hold even against quantum adversaries.

Proof. Any poly-time adversary makes at most $q = q(\lambda)$ 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

  $$(\mathsf{crs}, \mathsf{pk}^*) \leftarrow \mathsf{Sim}(1^\lambda, (\mathsf{pk}_1, \mathsf{pk}_2, c_1^*, c_{2,1}^*, c_{2,2}^*), \mathcal{L})$$

  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 9.3.6. *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 9.3.7. *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 9.3.8. *If the* diO-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 diO-CD. It suffices to show that the adversary cannot even find $c_{2,2}^*$.

Claim 9.3.9. *Assuming the properties from claim 9.3.8, for any* (m_0, m_1) *and any function* f*, we have*

$$\Pr\left[\mathcal{A}(C_0, C_1, \mathsf{aux}) = c^*_{2,2} : \begin{array}{c} (\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{KeyGen}_{\mathsf{PKE}}(1^\lambda), \\ c^*_{2,2} \leftarrow \{0,1\}^n, \\ c^*_1 \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m_0), c^*_{2,1} \leftarrow \mathsf{Enc}(\mathsf{pk}_2, m_1 \oplus c^*_{2,2}), \\ C_0 = \mathsf{FEInner}(f, \mathsf{sk}_1), \\ C_1 = \mathsf{Hyb}_3\mathsf{Inner}(f, \mathsf{sk}_1, m_1, c^*_1, c^*_{2,1}, g(c^*_{2,2})) \\ \mathsf{aux} = (\mathsf{pk}_1, \mathsf{pk}_2, m_0, m_1, f) \end{array}\right] = \mathsf{negl}(\lambda)$$

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 $\Pr[\mathcal{A}(C_0, C_1, \mathsf{aux}) = c^*_{2,2}] = \mathsf{negl}(\lambda)$ 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 diO-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 9.3.10. *If the* diO-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 $\bot$.
- 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 9.3.11. *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 9.3.12. *If the* diO-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 $\bot$.
- 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 9.3.13. *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. □

9.4 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 9.4.1 (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.*

- $\mathsf{Gen}(1^\lambda, C)$ *takes in the security parameter and a circuit $C \in C$, 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,*

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

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

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

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 9.4.2 (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* $\mathsf{Adv} = (\mathsf{Adv}_1, \mathsf{Adv}_2)$ *where* Adv_1 *outputs a bipartite state on registers* R_1 *and* R_2*, the following holds:*

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

If this holds when vk *is also given to* Adv_1 *and* Adv_2*, then we say it has publicly verifiability. If this holds when* Adv_2 *is computationally unbounded, we say it has statistical returns.*

We note that unlike in diO, Adv does not receive additional auxiliary input after the circuit C is sampled.

Theorem 9.4.3. *Assuming* $\mathsf{diO\text{-}CD}$*, there exists secure software leasing with (publicly verifiable)* β*-perfect strong finite-term security for all differing inputs circuits families, where* $\beta = 1/2 + \mathsf{negl}(\lambda)$*. Furthermore, it has statistical returns.*

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

- $\mathsf{SSL.Gen}(1^\lambda, C)$: Run $\mathsf{diO\text{-}CD.Obf}(1^\lambda, C)$ then output the result
- $\mathsf{SSL.Eval}(\widetilde{C}, x)$: Run $\mathsf{diO\text{-}CD.Eval}(\widetilde{C}, x)$ and outputs the result.
- $\mathsf{SSL.Verify}(\mathsf{vk}, \widetilde{C})$: We first describe the scheme with measurements. Evaluate $\mathsf{cert} \leftarrow \mathsf{diO\text{-}CD.Del}(\widetilde{C})$. Output $\mathsf{diO\text{-}CD.Verify}(\mathsf{vk}, \mathsf{cert})$. 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 $\mathsf{diO\text{-}CD}$. 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 $\rho(b)$.[5] 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 $\mathsf{diO\text{-}CD}$ and convexity of trace distance, we have $\mathsf{TD}(\rho'(0), \rho'(1)) = \mathsf{negl}(\lambda)$ whenever the certificate is valid.

We now argue that the probability $\rho'(b)$ can be used to evaluate $C_b(y^*)$ is at most $1/2 + \mathsf{negl}(\lambda)$ for any differing input y^*. Say that $\mathsf{Adv}_2(\rho'(b), x)$ outputs $C_b(x)$ with probability β_0 for all x. Then $\mathsf{Adv}_2(\rho'(1-b), x)$ also outputs $C_b(x)$ with probability $\geq \beta_0 - \mathsf{negl}(\lambda)$. Therefore, for any differing input y^*, $\mathsf{Adv}_2(\rho'(1-b), y^*)$ is incorrect with probability at least $1 - \beta_0 - \mathsf{negl}(\lambda)$. Thus, the probability of Adv_2 correctly evaluating for a random b is at most $1/2\beta_0 + 1/2(1 - (\beta_0 - \mathsf{negl}(\lambda))) = 1/2 + \mathsf{negl}(\lambda)$. In other words, $\beta \leq 1/2 + \mathsf{negl}(\lambda)$. □

[5] 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.

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

Theorem 9.4.4. *Assuming* $\mathsf{diO\text{-}CD}$*, there exists secure software leasing with (publicly verifiable)* β*-perfect strong finite-term security for all differing inputs circuits families* $\mathcal{C}$ *associated with sampler* $\mathcal{D}_{\mathcal{C}}$*, where*

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

Furthermore, it has statistical returns, where β *is taken as the maximum over unbounded* Adv*.*

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 $\mathsf{Adv}_2(\rho'(1), y^*)$ outputs $C_0(y^*)$ with probability at most β for every differing input y^*. Therefore $\mathsf{Adv}_2(\rho'(0), y^*)$ outputs $C_0(y^*)$ with probability at most $\beta + \mathsf{negl}(\lambda)$. □

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 iO.

Corollary 9.4.5. *Assuming post-quantum one-way functions and* $\mathsf{diO\text{-}CD}$*, there exists strong secure software leasing for pseudorandom functions, evasive functions, random point functions, and compute-and-compare circuits.*

Sketch. We sketch the result for psuedorandom functions and note that the other classes can be argued similarly. Because of Theorem 9.4.4, it suffices to construct a differing-inputs circuit family. 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 iO [BLW17], which is implied by $\mathsf{diO\text{-}CD}$. Then $\{(C_k, C_{k^*_y}) : y^* \leftarrow \{0,1\}^\lambda\}$ is a differing inputs circuits class where

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

□

Part II

Secret Sharing with Certified Deletion

In this part, we show how to equip secret sharing with certified deletion. Secret sharing [Sha79, Bla79, ISN87] is a foundational cryptographic primitive that allows a dealer to distribute a secret s among n parties so that only certain "authorized" subsets of the parties may recover the secret. A particularly appealing aspect of secret sharing as compared to most other cryptographic primitives is it *doesn't require computational hardness assumptions*. That is, one can construct secret sharing secure against any computationally unbounded adversary, for any monotone access structure (e.g. [ISN87, BL90, LV18]).

However, the security of these schemes still rests on a stringent assumption: over the course of the (potentially unbounded) adversary's operation, it only ever sees an unauthorized set of shares. This may be unacceptable for users sharing particularly sensitive information. Even if an adversary initially may only access a limited number of shares, over time they may be able to corrupt more and more parties, or perhaps more and more shares become compromised independently and are leaked into the public domain. A user who becomes paranoid about this possibility generally has no recourse, and, worse yet, cannot even *detect* if an adversary has obtained access to enough shares to reconstruct their secret.

By making use of uniquely quantum effects, secret sharing with certified deletion enables security *even against adversaries that eventually corrupt an authorized set of shares*.

Previously, Bartusek and Khurana [BK23] defined and constructed a very limited flavor of secret sharing with certified deletion: 2-out-of-2 secret sharing where only one the two shares can be deleted. We show that it is possible to introduce certified deletion guarantees into more versatile and general flavors of secret sharing.

Chapter 10

Results

We formulate two powerful but incomparable notions of certified deletion security for general-purpose secret sharing schemes, and show how to construct a scheme satisfying each definition. One of our key technical tools is a high-rate seedless extractor from certain quantum sources of entropy that significantly generalizes and improves upon the "XOR extractor" of [ABKK23].

No-signaling security. First, we address the shortcomings of [BK23]'s security definition for 2-out-of-2 secret sharing sketched above, and formulate a natural extension that (i) applies to schemes for *any* monotone access structure,[1] and (ii) allows for the possibility that *any* of the shares may be deleted.

Consider a scenario involving multiple non-communicating adversaries that each individually can access some unauthorized set of shares. These adversaries may share entanglement, but may not exchange messages. Now, the user may request that some of its shares are deleted. If the adversaries jointly delete enough shares so that the remaining undeleted shares form an *unauthorized* set, then we require that the user's secret remains private even given the combined views of all the adversaries. That is, even if a single adversarial entity can eventually corrupt *all* of the parties, the secret is still hidden if enough shares have previously been deleted.[2]

We refer to this security notion for secret sharing schemes as *no-signaling security* (see chapter 14 for a precise definition), emphasizing the fact that shares must be deleted by adversaries that cannot yet pool information about an authorized set of shares, as this would trivially allow for reconstruction of the secret. Then, in chapter 15 we show how to combine [BK23]'s simple 2-out-of-2 secret sharing scheme with any standard secret sharing scheme for monotone access structure $\mathbb{S}$ (e.g. [ISN87, BL90, LV18]) in order to obtain a secret sharing scheme for $\mathbb{S}$ with no-signaling security.

Theorem 10.0.1 (Informal)**.** *There exists a secret sharing scheme with no-signaling certified deletion security for any monotone access structure* $\mathbb{S}$*.*

Adaptive security. Next, we consider a particularly cunning but natural class of adversaries. Suppose that initially the adversary only obtains access to some unauthorized set of shares. At some point, the user becomes paranoid and requests that some subset of these shares are deleted. The adversary obliges but then continues to corrupt new parties or locate other leaked shares. The adversary may continue to delete some of these shares to appease the user, while continuing to work behind the scenes to mount a

[1] A monotone access structure defines sets of authorized users by a set $\mathbb{S}$ of subsets of n where for any subset $S \in \mathbb{S}$, it holds that $S' \in \mathbb{S}$ for all supersets $S' \supset S$.

[2] We remark that this definition also captures adversaries that don't end up corrupting *all* the shares, by imagining that there is a separate component of the adversary that honestly deletes the uncorrupted shares.

long-term attack on the system. However, as long as the set of corrupted parties minus the set of certifiably deleted shares continues to be unauthorized, we can hope that the user's secret remains private from such an adversary.

Unfortunately, the notion of no-signaling security does not capture such *adaptive* behavior. No-signaling security only models adversaries that delete once, and then receive some extra information after this single round of deletion. Thus, we formalize *adaptive security* as an alternative and quite strong notion of certified deletion security for secret sharing schemes (see chapter 14 for a precise definition).

Protecting against such arbitrarily adaptive adversaries turns out to be a significant challenge. The main technical component of this part realizes a secret sharing scheme with adaptive certified deletion security for the specific case of threshold access structures (chapter 16).

Theorem 10.0.2 (Informal)**.** *There exists a threshold secret sharing scheme with adaptive certified deletion security.*

High-rate seedless extractors from quantum sources of entropy. One of our technical building blocks is an improved method for seedless extraction from certain quantum sources of entropy. Roughly, the source of entropy comes from performing a standard basis measurement on a register that is in superposition over a limited number of Fourier basis states.

While seedless extraction from such sources of entropy [ABKK23] has been a crucial component in previous realizations of cryptographic primitives with certified deletion [BK23],[3] the technique had been limited to (i) extracting from qubit registers (i.e. where data is natively represented as superpositions of bitstrings) and (ii) extracting only a single bit of entropy. Here, we generalize these techniques to extract from qu*dit* registers (i.e. where data is natively represented as superpositions of vectors over finite fields), and produce several field elements worth of entropy, vastly improving the rate of extraction. Beyond being interesting in its own right, these improvements are crucial for showing security of our construction of threshold sharing with adaptive certified deletion security. Moreover, we show how to apply these high-rate extraction techniques to extension fields. This allows us to represent our quantum shares as string of qubits (as opposed to qudits), removing the need for entanglement in our construction. We refer the reader to section 11.3 and chapter 13 for more details.

[3] See discussion therein for why *seedless* as opposed to seeded extraction is crucial.

Chapter 11

Technical Overview

Intuitively, certified deletion for secret sharing aims to keep the secret private from an adversary if the total set of undeleted shares they have access to is unauthorized. One could formalize this by considering an adversary who initially receives an unauthorized set of shares and then deletes some of them. If the undeleted shares are still unauthorized when combined with the shares that the adversary did not receive, then we allow the adversary to access these remaining shares. This closely matches the definition of encryption with certified deletion, where the adversary initially receives and deletes a ciphertext $\mathsf{Enc}(k, m)$ encrypting message m using key k, and then later receives the key k.

However, this definition is not meaningful for all access structures. For example, in a k out of n access structure where $k < n/2$, the shares that the adversary does not start with *already* form an authorized set on their own, so it never makes sense to allow the adversary to access all of these shares at once. In this section, we give an overview of two different ways to address this definitional deficiency: no-signaling certified deletion and adaptive certified deletion.

11.1 No-Signaling Certified Deletion

In no-signaling certified deletion, we address this problem by allowing the adversary to delete from multiple sets of shares $P_1, \ldots, P_\ell$. However, if $P_1 \cup \cdots \cup P_\ell$ contains all shares, then the adversary as a whole gets to see every share before it generates any deletion certificates. Thus, to prevent trivial attacks, we do not allow the adversary to communicate across sets. However, the different parts of the adversary may still *share entanglement*. This modification yields the no-signaling certified deletion game $\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(s)$ for secret s and access structure $\mathbb{S}$ over n parties, which we describe here.

1. The challenger secret-splits s into n shares with access structure $\mathbb{S}$.

2. Each adversary Adv_i is initialized with one register of a shared state $|\psi\rangle$, receives the shares in a set P_i, and produces some set of certificates $\{\mathsf{cert}_j\}_{j \in P_i}$. If Adv_i does not wish to delete share j, then it may set $\mathsf{cert}_j = \bot$.

3. If the total set of shares that have not been deleted is unauthorized, then output the joint view of the adversaries. Otherwise, output $\bot$.

No-signaling certified deletion for secret sharing requires that for every secret pair (s_0, s_1) and every partition $P = (P_1, \ldots, P_\ell)$ of $[n]$, the outputs of $\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(s_0)$ and $\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(s_1)$ have negligible trace distance.

Tool: 2-of-2 Secret Sharing with Certified Deletion [BK23]. Recently, Bartusek and Khurana constructed a variety of primitives with certified deletion. One of these primitives is a secret sharing scheme which splits a secret s into a quantum share $|\mathsf{sh}_1\rangle$ and a classical share sh_2, along with a verification key vk that can be used to test the validity of deletion certificates. Given either one of the shares, the secret is hidden. Furthermore, if an adversary given $|\mathsf{sh}_1\rangle$ performs a destructive measurement that yields a valid deletion certificate, then they will never be able to recover s, even if they later obtain sh_2. Note that in this scheme, only one of the two shares can be deleted.

A Black-Box Compiler. We show how to compile Bartusek and Khurana's 2-of-2 certified deletion scheme together with any classical secret sharing scheme into a secret sharing scheme with no-signaling certified deletion. Notably, the compiled scheme inherits the same access structure as the classical secret sharing scheme. Thus, one can construct secret sharing with no-signaling certified deletion for general access structures by using any classical secret sharing scheme for general access structures, e.g. [ISN87, ABF+19].

As a starting point, let us first construct a scheme where only one of the shares can be deleted.

1. Secret split the secret s into a quantum share $|\mathsf{qsh}\rangle$ and a classical share csh using the 2-of-2 secret sharing scheme with certified deletion. This also produces a verification key vk.
2. Split the 2-of-2 classical share csh into classical shares $\mathsf{csh}_1, \ldots, \mathsf{csh}_n$ using the classical secret sharing scheme for $\mathbb{S}$.
3. The verification key is vk and the i'th share is csh_i. The deletable quantum share is $|\mathsf{qsh}\rangle$.

Given the quantum share and any authorized set of classical shares, s can be reconstructed by first recovering csh from the authorized set. On the other hand, any adversary which attempts to delete $|\mathsf{qsh}\rangle$ with only access to an unauthorized set of classical shares has no information about the 2-of-2 classical share csh. Thus if they produce a valid deletion certificate, they will have no information about s even after obtaining the rest of the classical shares, which only reveals csh.

Who Deletes? To finish the compiler, we need to enable certified deletion of *any share*. This can be achieved by adding a step at the beginning of the compiler to create n classical shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ of s with the same access structure $\mathbb{S}$. Then, the splitter can enable certified deletion for each share sh_i by using the prior compiler to produce a set of classical shares $\mathsf{csh}_{i,1}, \ldots, \mathsf{csh}_{i,n}$, a deletable quantum share $|\mathsf{qsh}_i\rangle$, and a verification key vk_i. The i'th share contains the deletable state $|\mathsf{qsh}_i\rangle$, as well as $\{\mathsf{csh}_{j,i}\}_{j\in[n]}$.

Note that anyone holding share i is able to delete sh_i by deleting $|\mathsf{qsh}_i\rangle$, as discussed previously. If sufficiently many shares are deleted, so that the only remaining sh_i form an unauthorized set, then no adversary can learn anything about the secret even after obtaining all of the remaining shares and residual states.

Proof of Security: Guessing Deletions. Although the intuition is straightforward, there is a nuance in the proof of security. When proving security, we wish to replace the deleted 2-of-2 secrets sh_i with empty secrets $\bot$. If we could do so, then security immediately reduces to the security of the classical $\mathbb{S}$-scheme, since only an unauthorized set of sh_i remains. However, it is difficult to determine which of these 2-of-2 secrets sh_i to replace with $\bot$ when preparing the shares.

Since non-local operations commute, we could consider generating the shares for each adversary Adv_i one at a time. For example, supposing Adv_1 operates on the set of shares $P_1 \subset [n]$, the experiment could initialize Adv_1 with uniformly random shares, and then for each $i \in P_1$, reverse-sample the shares

$\{\mathsf{csh}_{i,j}\}_{j\in[n]\setminus P_1}$ for the rest of the adversaries to match either sh_i or $\bot$, depending on whether or not Adv_1 deleted share i.

Unfortunately, we cannot continue this strategy for all of the adversaries. It may be the case that the union of Adv_1 and Adv_2's shares $P_1 \cup P_2$ contains an authorized set. Thus, when initializing Adv_2, the challenger must already know whether, for each $i \in P_2$, the i'th share of s should be set to sh_i or $\bot$ (since this will be determined by $\{\mathsf{csh}_{i,j}\}_{j\in P_1\cup P_2}$). This view is constructed before the adversary decides whether or not to delete share i, so the only way for the challenger to do this is to guess whether the adversary will delete share i or not.

Now, guessing which shares the entire set of adversaries will delete incurs a multiplicative exponential (in n) loss in security. Fortunately, Bartusek and Khurana's 2-of-2 scheme actually satisfies an *inverse exponential* trace distance between the adversary's view of any two secrets, after deletion. Thus, by setting the parameters carefully, we can tolerate this exponential loss from guessing, and show that our scheme for general access structures satisfies negligible security.

11.2 Adaptive Certified Deletion

Intuitively, any definition of certified deletion should allow the adversary to eventually receive an authorized set of shares, as long as they have previously deleted enough shares so that their total set of undeleted shares remains unauthorized. In no-signaling certified deletion, we allowed multiple non-communicating adversaries to delete from different unauthorized sets of shares. That is, when Adv_i generates its set of certificates, it only has access to a single unauthorized set P_i. However, one could also imagine a more demanding setting where, after deleting some shares, the adversary can adaptively corrupt *new shares*, as long as their total set of undeleted shares remains unauthorized. Then, they can continue deleting shares and corrupting new shares as long as this invariant holds. This setting arises naturally when we consider an adversary which covertly compromises shares over a long period of time, while occasionally deleting shares to avoid revealing the extent of the infiltration. We call this notion *adaptive certified deletion*. It is defined using the following adaptive certified deletion game $\mathsf{SS\text{-}ACD}_{\mathbb{S}}(s)$.

1. The challenger splits the secret s into n shares with access structure $\mathbb{S}$. The adversary starts with an empty corruption set C and an empty deletion set D.

2. For as many rounds as the adversary likes, it gets to see the shares in C and choose whether to corrupt or delete a new share.

   Corrupt a new share. The adversary corrupts a new share i by adding i to C. If $C\setminus D$ is authorized, the experiment immediately outputs $\bot$.

   Delete a share. The adversary outputs a certificate cert for a share i. If cert is valid, add i to D. Otherwise, the experiment immediately outputs $\bot$.

3. When the adversary is done, the experiment outputs its view.

Adaptive certified deletion for secret sharing requires that for every secret pair (s_0, s_1), the outputs of $\mathsf{SS\text{-}ACD}_{\mathbb{S}}(s_0)$ and $\mathsf{SS\text{-}ACD}_{\mathbb{S}}(s_1)$ have negligible trace distance. In this work, we focus on the (k, n) threshold access structure, where any set of size $\geq k$ is authorized.

Incomparable Definitions. We have already seen that no-signaling certified deletion does not imply adaptive certified deletion. It is also the case that adaptive certified deletion does not imply no-signaling certified deletion. Consider a two-part no-signaling adversary Adv_1 and Adv_2 with views P_1 and P_2. To

change $(\mathsf{Adv}_1, \mathsf{Adv}_2)$ to an adaptive adversary, one would need to come up with a transformation that deletes the same shares as $(\mathsf{Adv}_1, \mathsf{Adv}_2)$, in the same way. However, Adv_1 might not even decide which shares to delete until after they have seen every share in P_1. So, the new adaptive adversary would have to corrupt all of P_1 before it can delete a single share that Adv_1 would. Similarly, it would also have to corrupt all of P_2 before it knows which shares Adv_2 would delete. However, if $P_1 \cup P_2$ is authorized, then the experiment would abort before the new adaptive adversary gets the chance to delete shares for both Adv_1 and Adv_2.

An Attack on the No-Signaling Construction. Unfortunately, the previous construction actually does *not* in general satisfy adaptive certified deletion security. Indeed, observe that the classical parts of each share can never be deleted. Because of this, an adversary could, for any i, obtain k classical shares $\mathsf{csh}_{i,1}, \ldots, \mathsf{csh}_{i,k}$ that reveal csh_i, simply by corrupting and immediately deleting the first k shares one-by-one. Afterwards, the adversary will always have *both* the classical share csh_i and the quantum share $|\mathsf{qsh}_i\rangle$ when it corrupts a new share i, so it can recover the underlying classical share sh_i. Now it can "delete" the i'th share while keeping sh_i in its memory. Eventually, it can collect enough sh_i in order to obtain the secret s.

The core problem with the no-signaling construction is the fact that it encodes the 2-of-2 classical shares csh in a form which can never be deleted. If we were to take a closer look at Bartusek and Khurana's 2-of-2 scheme, we would observe that csh contains a mapping θ of which parts of $|\mathsf{qsh}\rangle$ encode the secret (the "data indices") and which parts contain only dummy information used to verify certificates (the "check indices"). Without θ, there is no way to decode the secret. Unfortunately, in order to encode θ in a deletable form, we seem to be back where we started - we need secret sharing with adaptive certified deletion!

A New Construction. To avoid this pitfall, we take a new approach that allows the parties to reconstruct *without knowledge of the check indices*. This removes the need to encode θ altogether. To achieve this, we begin with Shamir's secret sharing [Sha79], in which the shares are evaluation points of a degree $k-1$ polynomial f where $f(0) = s$. This polynomial is over some finite field $\mathbb{K}$ with at least $n+1$ elements. A useful property of Shamir's secret sharing is that it has good error-correcting properties - in fact, it also forms a Reed-Solomon code, which has the maximum possible error correction [RS60].

To split a secret s, we start by constructing a polynomial f where $f(0) = s$. Each share contains some number of evaluations of f encoded in the computational basis. These evaluations are mixed with a small number of random Fourier basis states that will aid in verifying deletion certificates. The positions of these checks, along with the value encoded, make up the verification key. An example share and key are illustrated here.

$$\begin{array}{lccccccccccc} \mathsf{sh} = & |f(1)\rangle & \otimes & |f(2)\rangle & \otimes & \mathsf{QFT}\,|r_1\rangle & \otimes & |f(4)\rangle & \otimes & \mathsf{QFT}\,|r_2\rangle & \otimes & \ldots \\ \mathsf{vk} = & * & & * & & r_1 & & * & & r_2 & & \ldots \end{array}$$

When reconstructing the secret, these checks are essentially random errors in the polynomial evaluation. By carefully tuning the degree of the polynomial together with the number of evaluations and checks in each share, we can ensure that any k shares contain enough evaluation points to correct the errors from the check positions, but that any $k-1$ shares do not contain enough evaluation points to determine the polynomial. This results in f being determined by slightly more than $k-1$ shares worth of evaluations. Additionally, we slightly increase the degree of the polynomial to account for the limited amount of information that an adversary can retain after deletion. See Chapter 16 for more details.

Share deletion and certificate verification follow the established formula. To delete a share, measure it in the Fourier basis and output the result as the certificate. To verify the certificate, check that it matches the verification key at the check positions.

Proving Adaptive Certified Deletion. Intuitively, we want to show that after the adversary deletes a share, the next share it corrupts gives it no additional information, no matter how many shares the adversary has seen so far. To formalize this, we will show that the adversary cannot distinguish between the real $\mathsf{SS\text{-}ACD}_{(k,n)}(s)$ experiment and an experiment in which each share is generated uniformly at random and independently of the others. Since the first $k-1$ shares to be corrupted do not yet uniquely determine the polynomial f, they already satisfy this. Thus, we can restrict our attention to modifying the last $n-k+1$ shares to be corrupted.

It will be useful to name the shares in the order they are corrupted or deleted. The a'th share to be corrupted is c_a, and the b'th share to be deleted is share d_b. In the (k,n) threshold case, if c_{k-1+b} is corrupted before d_b is deleted, then $C \backslash D$ has size k and is authorized, so the experiment will abort.

Techniques from BK23. We begin by recalling the techniques introduced in [BK23] to analyze 2-of-2 secret sharing with certified deletion, along with the construction. These techniques will form the starting point of our proof. To share a single-bit secret $s \in \{0,1\}$, sample random $x, \theta \leftarrow \{0,1\}^\lambda$ and output

$$\mathsf{sh}_1 = H^\theta \left|x\right\rangle, \quad \mathsf{sh}_2 = \left(\theta, s \oplus \bigoplus_{i:\theta_i=0} x_i\right), \quad \mathsf{vk} = (x,\theta),$$

where H^θ denotes applying the Hadamard gate H to the i'th register for each $i : \theta_i = 1$. Bartusek and Khurana showed that if an adversary given sh_1 produces a certificate cert such that $\mathsf{cert}_i = x_i$ for every check position $i : \theta_i = 1$, then they cannot distinguish whether $s = 0$ or $s = 1$ even if they later receive sh_2. Their approach has three main steps.

1. First, they delay the dependence of the experiment on s by initializing sh_1 to be the register $\mathcal{X}$ in $\sum_x \left|x\right\rangle^{\mathcal{X}} \otimes \left|x\right\rangle^{\mathcal{Y}}$. Later, the challenger can obtain x by measuring register $\mathcal{Y}$, and use it to derive s.

2. Second, they argue that if the adversary produces a valid deletion certificate, then sh_1 has been "almost entirely deleted", in the sense that the challenger's copy satisfies a checkable predicate with high probability. Intuitively, this predicate shows that the data positions ($\theta_i = 0$) of the challenger's copy have high joint entropy when measured in the computational basis. To show that the predicate holds, they use the fact that the adversary does not have sh_2 in their view, so sh_1 looks uniformly random. This allows a cut-and-choose argument where the locations of the check indices are determined *after* the adversary outputs its deletion certificate.

3. Finally, they show that the challenger derives a bit s that is uniformly random and independent of the adversary's view. This utilizes a result from [ABKK23] which shows that XOR is a good seedless extractor for entropy sources that satisfy the aforementioned predicate.

Adapting to Secret Sharing. As a starting point, let us try to adapt these techniques to undetectably change shares to uniformly random. For concreteness, consider the task of switching a share c_{k-1+b} to uniformly random. Although we have not yet outlined the general proof structure, we will eventually need to perform this task. We will use this starting point to gain insights that will help guide the eventual proof structure.

1. The first step is to delay the synthesis of the secret information until after the adversary outputs a deletion certificate. In our case, we will delay creating share c_{k-1+b} until after the adversary produces a valid certificate for share d_b.

   This can be achieved by sampling the first $k-1$ corrupted shares to be uniformly random, then using polynomial interpolation to prepare the rest of the shares. More concretely, consider the first corrupted $k-1$ shares $c_1, \ldots, c_{k-1}$. The challenger will prepare each of these shares c_a using two registers $\mathcal{C}_{c_a}$ and $\mathcal{S}_{c_a}$, then send the share to the adversary in register $\mathcal{S}_{c_a}$. To prepare the j'th position of c_a, the experiment challenger prepares either a uniform superposition $\sum_{x\in\mathbb{K}} |x\rangle^{\mathcal{C}_{c_a,j}}$ or $\sum_{x\in\mathbb{K}} \mathsf{QFT}\,|x\rangle^{\mathcal{C}_{c_a,j}}$, depending on whether j is an evaluation position or a check position. If j is an evaluation position for share c_a, the experiment challenger copies $\mathcal{C}_{c_a,j}$ to $\mathcal{S}_{c_a,j}$ in the computational basis, yielding

   $$\propto \sum_{x\in\mathbb{K}} |x\rangle^{\mathcal{S}_{c_a,j}} \otimes |x\rangle^{\mathcal{C}_{c_a,j}},$$

   and otherwise it copies $\mathcal{C}_{c_a,j}$ to $\mathcal{S}_{c_a,j}$ in the Fourier basis, yielding

   $$\propto \sum_{x\in\mathbb{K}} \mathsf{QFT}\,|x\rangle^{\mathcal{S}_{c_a,j}} \otimes \mathsf{QFT}\,|x\rangle^{\mathcal{C}_{c_a,j}}.$$

   Note that the adversary cannot determine which positions are evaluation positions and which are check positions, since each $\mathcal{S}_{c_a,j}$ register looks maximally mixed. Also observe that $\mathcal{C}_{c_a}$ contains a copy of share c_a, and the evaluation positions in the initial $k-1$ $\mathcal{C}_{c_a}$ registers determine the polynomial f. Then, when the adversary requests to corrupt a later share, the challenger computes the evaluation points for that share by performing a polynomial interpolation using its copies of the prior shares. For reasons that will become apparent shortly, we require that share d_b is included when interpolating c_{k-1+b}. The other points may be arbitrary.

   The above procedure is actually slightly simplified; since the degree of f is slightly larger than the number of evaluation positions in $k-1$ shares, the first $k-1$ shares do not quite determine f. To remedy this, we will also initialize a small portion of *every* $\mathcal{S}_i$ to be uniformly random, before any interpolation takes place.

2. Next, we will need to show that $\mathcal{C}_{d_b}$, which contains the challenger's copy of share d_b, satisfies the deletion predicate if cert_{d_b} passes verification. This is not hard to show if d_b was generated uniformly at random, but it is not clear what happens if the adversary has some information about where the check positions are in d_b before deleting it. The first $k-1$ shares are uniformly random in the original experiment, so this is not a problem for any share which is deleted before c_k is corrupted. However, later shares depend on earlier shares, potentially leaking information about the check positions. This will be our first barrier to overcome.

3. Finally, we will need to show that interpolating c_{k-1+b} using $\mathcal{C}_{d_b}$ produces a uniformly random value whenever $\mathcal{C}_{d_b}$ satisfies the deletion predicate. In other words, **polynomial interpolation should double as a good randomness extractor from deleted shares**. Fortunately, polynomial interpolation is a matrix multiplication, and we have intuition from the classical setting that linear operations are good randomness extractors. Since a small amount of every share is uniformly random "for free", the extractor needs to produce only slightly less than a full share's worth of evaluations to produce c_{k-1+b}. This is our second technical contribution, which we will revisit in Section 11.3.

In step two, we seem to need the evaluation points in d_b to look like the check positions when d_b is deleted, i.e. they should be uniformly random and independent of the rest of the adversary's view. A natural approach to ensure this is to modify the shares to uniformly random round-by-round over a series of hybrid experiments. In hybrid i, the first $k-1+i$ shares to be corrupted are uniformly random. Since d_{i+1} must be deleted before c_{k+i} is corrupted (or else the experiment aborts), d_{i+1} must have been one of the uniformly random shares. Now we can apply the cut-and-choose argument to show that $\mathcal{C}_{d_i}$ satisfies the deletion predicate, thereby satisfying the extractor requirements to change c_{k+i} to be uniformly random and reach hybrid $i+1$. The first $k-1$ shares are already uniformly random, which gives us an opening to begin making modifications in round k.

Unfortunately, the strategy of modifying the shares one-by-one to be uniformly random has a major flaw. In particular, the challenger needs to produce additional polynomial evaluations whenever the adversary wishes to corrupt another share, which it does via interpolation. Recall that in order to claim that c_{k-1+i} is indistinguishable from random, we apply an extractor which uses register $\mathcal{C}_{d_i}$ as its source of entropy. But in order to invoke the security of the extractor, it seems that we cannot allow the challenger to re-use $\mathcal{C}_{d_i}$ when interpolating later points, as this might leak additional information about the source to the adversary.

To get around this issue, we might require that the challenger never uses $\mathcal{C}_{d_i}$ again to interpolate later points. However, the randomness extractor outputs *less* randomness than the size of a share. Intuitively, this occurs because the adversary can avoid fully deleting the source share d_i by guessing the location of a very small number of check positions.[1] Imperfect deletion limits the entropy of the source, which in turn limits the size of the extractor output. Now, since the challenger started with *exactly* enough evaluations to uniquely determine f, if we take away the points in $\mathcal{C}_{d_i}$ then there are no longer enough evaluation points remaining to create the rest of the shares, even given the newly interpolated points.

Predicates First, Replacement Later. Intuitively, the problem outlined above arises from the possibility that the adversary receives additional information about earlier shares from the later ones, since they are all correlated through the definition of the polynomial f. Our first idea to overcome this issue is to prove that the predicate holds for all rounds *before* switching any shares to uniformly random. In particular, we will consider a sequence of hybrid experiments where in the i'th hybrid, the challenger performs the predicate measurement on $\mathcal{C}_{d_i}$ after receiving and verifying the corresponding certificate. If the measurement rejects, the experiment immediately aborts and outputs $\perp$.

If we can undetectably reach the last hybrid experiment, then it is possible to undetectably replace every share with uniform randomness by working backwards. In the last hybrid experiment, either the predicate holds on the challenger's copy $\mathcal{C}_{d_{n-k+1}}$ of share d_{n-k+1} or the experiment aborts. In either case, the last share c_n to be corrupted can be undetectably replaced with uniform randomness. Since no further shares are interpolated, we no longer run into the issue of re-using the randomness source $\mathcal{C}_{d_{n-k+1}}$, allowing the challenger to safely complete the experiment. Then, once c_n is uniformly random, the challenger no longer needs to interpolate shares after c_{n-1}, so c_{n-1} can also be replaced with uniform randomness. This argument can be continued until all shares are replaced.

To undetectably transition from hybrid i to hybrid $i+1$, we must show that the predicate measurement returns success with high probability on $\mathcal{C}_{d_{i+1}}$. This is not hard to show for shares which are deleted *before* the k'th share is corrupted, because the deleted shares must be one of the shares which were generated

[1] One may wonder whether it is possible to instead use coset states for the shares, which provide guarantees of *full* deletion [BGK+24]. Unfortunately, coset states induce errors which are the sum of a small number of uniformly random vectors. It is not clear how to correct these errors to reconstruct the secret without prior knowledge of the underlying subspace. However, encoding the subspace brings us back to the original problem of secret sharing with adaptive certified deletion.

uniformly at random. However, it is not clear how to show for shares which are deleted *after* the k'th share is corrupted, since this seems to require replacing c_k with uniform randomness, which brings us back to our previous problem.

Chaining Deletions via Truncated Experiments. Our second insight is the observation that the result of a measurement made when d_i is deleted *is independent of later operations*. Thus, when arguing about the probability that the predicate measurement accepts on $\mathcal{C}_{d_i}$, it is sufficient to argue about the truncated experiment that ends immediately after the predicate measurement on $\mathcal{C}_{d_i}$. Crucially, the adversary cannot corrupt share c_{k+i} in the truncated experiment without causing the experiment to abort due to $|C \backslash D| \geq k$. Instead, c_{k-1+i} is the last share that can be corrupted. This prevents the catastrophic re-use of $\mathcal{C}_{d_i}$ after share c_{k-1+i} is constructed.

Let us assume that we have already shown that the deletion predicate measurement accepts on $\mathcal{C}_{d_i}$ with high probability; for example, this clearly holds for d_1, which must be corrupted before c_k is corrupted. How likely it is to accept on $\mathcal{C}_{d_{i+1}}$? Say the deletion predicate measurement accepts on $\mathcal{C}_{d_i}$. Then we can invoke the extractor to undetectably replace share c_{k-1+i} with uniform randomness in the truncated game, since no further shares are corrupted before the game ends. We can use similar logic to replace each of the first $k-1+i$ shares to be corrupted in the truncated game. At this point, the adversary has no choice but to delete a uniformly random share as d_{i+1}, so we can apply a cut-and-choose argument to show that the predicate holds with high probability on $\mathcal{C}_{d_{i+1}}$ This argument can be repeated inductively to show that the predicate holds in each of the polynomially many rounds.

Recap of the First Challenge. In summary, the first challenge to address in proving adaptive certified deletion is the possibility of later shares leaking information about prior shares through the re-use of $\mathcal{C}_{d_b}$ in interpolation. This prevents directly replacing each share with uniform randomness. To sidestep this issue, we first argue that every $\mathcal{C}_{d_b}$ is a good source of entropy using a series of games which end after d_b is deleted. Then even if $\mathcal{C}_{d_b}$ is used to interpolate both share c_{k-1+b} and c_{k+b}, we can rely on the entropy from $\mathcal{C}_{d_{b+1}}$ to mask its re-usage when interpolating c_{k+b}.

11.3 High Rate Seedless Extractors from Quantum Sources of Entropy

The final task to finish the proof of adaptive certified deletion security in the previous section is to show that polynomial interpolation is a good randomness extractor for entropy sources formed by deleted shares. Although polynomial interpolation arises quite naturally in our construction, there are additional technical reasons why it would be difficult to design a construction for adaptive certified deletion using existing extractors.

If we were to design a scheme using a *seeded* extractor, as done in [BI20b], then every deletion would need to be independent of the seed to avoid the entropy source depending on the seed. However, as we saw with the no-signaling construction, safely encoding the seed seems to already require secret sharing with adaptive certified deletion. [BK23] makes use of the seedless XOR extractor developed by [ABKK23] to avoid a similar problem. Unfortunately, the XOR extractor produces only a single bit from a relatively large input. In the case of threshold secret sharing, the extractor must use the randomness produced by deleting a single share to extract an output which is only slightly smaller than a full share.

To address this need, we give a new family of seedless randomness extractors for quantum entropy sources with high rate. These constructions have connections to linear error-correcting codes and may be of independent interest.

A Family of Extractors. The input of the extractor is a vector of M elements of a finite field $\mathbb{F}$, and the output is a vector of m elements of $\mathbb{F}$. The source consists of a register $\mathcal{X}$ which may be arbitrarily entangled with a side-information register $\mathcal{A}$. If the register $\mathcal{X}$ is in superposition over Fourier basis vectors with Hamming weight $\leq (M-m)/2$ in $\mathbb{F}$, then we can argue that the output $\mathsf{Extract}(\mathcal{X})$ is uniformly random, even given $\mathcal{A}$.[2]

The extractor family consists of matrices $R \in \mathbb{F}^{m \times M}$ such that every set of m columns of R are linearly independent. In other words, R is a parity check matrix for a linear error-correcting code with distance at least m. An extractor R is applied by coherently multiplying R with the source register $\mathcal{X}$ in the computational basis and writing the result to the output register.

Application to Polynomial Interpolation. This family generalizes both the XOR extractor and polynomial interpolation. The XOR extractor can be represented as the all-ones matrix with one row. Each column is non-zero, so the extractor can produce a one-bit output. In the case of polynomial interpolation, we can write the linear interpolation operator for a polynomial f as a matrix R with $\mathsf{deg}(f)+1$ columns and a number of rows equal to the number of points being interpolated. R is a sub-matrix of a parity check matrix for a Reed-Solomon code, so it falls into the new extractor family. In fact, our result shows that any subset of columns in a polynomial interpolation matrix forms a good randomness extractor for an appropriate randomness source. When interpolating a share c_{k-1+b}, we can write the interpolation matrix as $R = [R_1|R_2]$, where R_2 is applied to d_b and R_1 is applied to the other points x on the polynomial. Then the new share is $c_{k-1+b} = R_1 x + R_2 d_b$. If d_b has satisfies the deletion predicate, then our extractor result shows that $R_2 d_b$ is uniformly random. Thus, the newly interpolated share c_b is also uniformly random.

Removing Entanglement. A downside of using polynomials for secret sharing is that each evaluation point exists in field $\mathbb{F}$ whose size scales with the total number of distinct points that must be defined on the polynomial. For example, $\mathbb{F}$ might be $\mathbb{Z}_p$ for some prime $p > nt$, where t is the number of evaluations per share. Using the approach outlined so far, each check position must be encoded in the Fourier basis over the same field $\mathbb{K}$. However, a logical $\mathbb{Z}_p$-qudit requires $\lceil \log_2(nt+1) \rceil$ qubits, which must all be entangled to produce a Fourier basis element of $\mathbb{Z}_p$.

We show how to remove the entanglement of the construction to only use *single-qubit* states, either in the Hadamard basis or in the computational basis. We modify the construction by setting the field $\mathbb{F}$ to be the binary extension field with $2^{\lceil \log_2(nt+1) \rceil}$ elements, so that each check position consists of $\lceil \log_2(nt+1) \rceil$ qubits. Then, we *individually* set each of these qubits to be a random Hadamard basis element. The other parts of the construction remain the same. Note that computational basis vectors over $\mathbb{F}$ can be encoded as a tuple of computational basis qubits.

Proving the security of this modification requires an expansion of the extractor theorem to allow general finite fields $\mathbb{F}$, which may have p^k elements for some prime p and $k \geq 1$. A Fourier basis element for such a field is obtained by applying the quantum Fourier transform over the additive group of $\mathbb{F}$, which is $\mathbb{Z}_p^k$. In particular, a Fourier basis element of $\mathbb{F}$ consists of k Fourier basis elements of $\mathbb{Z}_p$. In the case where $p = 2$, these are single-qubit Hadamard basis elements.

We emphasize that the *only* change is to modify how Fourier basis elements are defined by allowing general finite fields; both the extractor family and the Hamming weight requirement remain the same (i.e. they are still defined with respect to $\mathbb{F}$, *not* $\mathbb{Z}_p$). To gain intuition about the usefulness of this statement, let us consider its application in our secret-sharing construction. Ideally, an honest deleter would measure each qubit of its share in the Hadamard basis. However, since the dealer can only verify check positions, which each consist of $\lceil \log_2(nt+1) \rceil$ qubits, we can only prove a bound on the Hamming weight of the

[2]The Hamming weight over a (potentially non-binary) finite field is being the number of nonzero entries in the vector.

deleted state over $\lceil \log_2(nt+1) \rceil$-sized chunks, which corresponds to $\mathbb{F}$. This matches the entropy source requirements of the theorem. On the other hand, the polynomial that the secret is encoded in is also over $\mathbb{F}$, so polynomial interpolation must take place over $\mathbb{F}$. This matches the extractor family.

11.4 Open Problems

Although our results significantly strengthen secret sharing to resist new classes of attacks, we have only scratched the surface of an area with many fascinating open problems. We mention a few of them here.

- **Adaptive Certified Deletion for General Access Structures.** Against adaptive attacks, we construct a secret sharing scheme for the special case of threshold access structures. Is it possible to construct one for *general access structures*?
- **Stronger Definitions.** We prove the security of our schemes against *either* "distributed" attacks (i.e. no-signaling security) *or* adaptive attacks. Can we (i) formulate natural security definitions that capture both types of attacks, and (ii) prove the security of secret sharing schemes under such all-encompassing definitions?
- **Other Threshold Primitives.** Aside from secret sharing, there are many other primitives which use thresholds or other access structures. For example, in threshold signatures, any k members may non-interactively sign messages under a secret key split between n parties [DF90]. Is it possible to construct *threshold signatures or other threshold primitives* with certified deletion?
- **High Rate Commitments with Certified Deletion.** A commitment with certified deletion allows the committed message to be certifiably and information-theoretically deleted [HMNY22, BK23]. However, current approaches either work in the random oracle model or require $\Theta(\lambda)$ qubits to commit to a single bit. Our new high-rate extractor (Theorem 13.0.1) provides a promising start to reduce the commitment overhead. Unfortunately, the proof technique pioneered by [BK23] for the plain model requires guessing the committed message, which incurs a security loss that is exponential in the size of the message. Is it possible to overcome this difficulty and construct commitments with certified deletion that are *not much larger than the committed message*?

Chapter 12

Preliminaries

12.1 Quantum Computation

A classical operation f can be applied to a quantum register $\mathcal{X}$ using the map

$$|x\rangle^{\mathcal{X}} \otimes |y\rangle^{\mathcal{Y}} \mapsto |x\rangle^{\mathcal{X}} \otimes |y + f(x)\rangle^{\mathcal{Y}}$$

We denote the application of this operation as $\mathcal{Y} \leftarrow f(\mathcal{X})$.

Quantum Fourier Transform over Finite Groups. Let G be a finite cyclic group and let $\omega_{|G|} := \exp(\frac{2\pi i}{|G|})$ be a primitive $|G|$'th root of unity. Let $\mathcal{X}$ be a register containing a G-qudits. The quantum Fourier transform (QFT) over G applied to $\mathcal{X}$ is the operation

$$|x\rangle^{\mathcal{X}} \mapsto \sum_{y \in G} \omega_{|G|}^{xy} |y\rangle^{\mathcal{X}}$$

Any Abelian group H may be decomposed as a product of cyclic groups $G_1 \times \cdots \times G_k$. The QFT over H is the tensor of the QFTs on each of these groups. For example, if $H = G^k$, then the QFT over H is given by the operation

$$|x\rangle^{\mathcal{X}} \mapsto \sum_{y \in G^k} \omega_{|G|}^{x \cdot y} |y\rangle^{\mathcal{X}}$$

When we consider taking a QFT over a finite field, we technically mean taking the QFT over its additive group. For example, if $\mathbb{F}$ has order p^k for some prime p, then its additive group is $\mathbb{Z}_p^k$ and the QFT is the mapping above, where $G = \mathbb{Z}_p$.

Fourier transforms are closely related to roots of unity. An n'th root of unity ω is an element of $\mathbb{C}$ such that $\omega^n = 1$. ω is a *primitive* n'th root of unity if $\omega^k \neq 1$ for every $k \in [n-1]$. The following claim about the summation of roots of unity will be useful.

Claim 12.1.1. *Let G be a cyclic group and let $\omega \neq 1$ be an $|G|$'th root of unity. Then*

$$\sum_{x \in G} \omega^x = 0$$

Proof.

$$\omega \sum_{x \in G} \omega^x = \sum_{x \in G} \omega^{1+x} \tag{12.1}$$

$$= \sum_{z \in G} \omega^z \tag{12.2}$$

where $z = 1 + x \in G$. Since $\omega \neq 1$ by definition, this can only be the case if $\sum_{x \in G} \omega^x = 0$. □

12.2 Statistics

Claim 12.2.1 (Hoeffding's Inequality [Hoe94])**.** *Let $X_1, \ldots, X_n$ be Boolean independent random variables. Let $\mu = \mathbb{E}[\sum_{i=1}^n X_i]$. Then for every $\delta > 0$,*

$$\Pr\left[\left|\sum_{i=1}^{n} X_i - \mu\right| \geq \delta\right] \leq 2\exp\left(\frac{-2\delta^2}{n}\right)$$

12.3 Polynomials and Reed-Solomon Codes

We give some useful facts about polynomials over finite fields and the related Reed-Solomon error-correcting code.

Interpolation. Lagrange interpolation gives a method of finding every point on a degree d polynomial f over any field $\mathbb{K}$, given any $d+1$ points on f [LM01]. In particular, for every degree d, there is a linear operation $\mathsf{Interpolate}_d(Y, X)$ that takes in a set of $d+1$ pairs $(x, f(x))$ and outputs $f(y)$ for each $y \in Y$. The operation to find a set of s points is described by a matrix $R \in \mathbb{K}^{s \times (d+1)}$.

Fact 12.3.1. *Let $R \in \mathbb{K}^{s \times (d+1)}$ be an interpolation matrix for a degree d polynomial. Then any s columns of R are linearly independent.*

Proof. Consider the matrix $P = [R| - I_s]$. It suffices to show that any s columns of P are linearly independent. Observe that $Py = 0$ if and only if y_i lies on the unique degree d polynomial defined by $y_1, \ldots, y_{d+1}$ for every $i \in [d+2, d+1+s]$. Assume, for the sake of contradiction, that some set of s columns of P were linearly dependent. Then there exists a y with exactly s non-zero values such that $Py = 0$. Since y is in the kernel of P, it consists of $d+1+s$ evaluations of a degree d polynomial. However, this would imply the existence of a degree d polynomial with $d+1+s-s = d+1$ zeros, which does not exist. □

Reed-Solomon Codes. A Reed-Solomon code is a error correcting code based on polynomials over a finite field $\mathbb{K}$ [RS60]. Given a message $m \in \mathbb{K}^d$, it encodes m as a polynomial f with degree $d+1$, then outputs s evaluations of f as the codeword. For finite field $\mathbb{K}$ and degree d, there exists an efficient correction procedure $\mathsf{Correct}_{\mathbb{K},d}(X)$ which attempts to recover a polynomial f using a noisy set of evaluation points X, e.g. [WB86, Gao03]. If X has size s and there are $< (s-d+1)/2$ pairs $(x, y) \in X$ such that $y \neq f(x)$, then $\mathsf{Correct}_{\mathbb{K},d}(X)$ outputs f.

12.4 Secret Sharing

Classical secret sharing. We introduce the standard notion of a secret sharing scheme, which allows one party to distribute a classical secret s among n parties, such that only certain subsets of parties have the ability to reconstruct the secret s. An access structure $\mathbb{S} \subseteq \mathcal{P}([n])$ for n parties is a monotonic set of sets, i.e. if $S \in \mathbb{S}$ and $S' \supset S$, then $S' \in \mathbb{S}$. Any set of parties $S \in \mathbb{S}$ is authorized to access the secret. Secret sharing for general monotone access structures was first introduced by [ISN87].

Definition 12.4.1 (Secret Sharing for Monotone Access Structures)**.** *A secret sharing scheme is specified by a monotone access structure $\mathbb{S}$ over n parties, and consists of two classical algorithms:*

- $\mathsf{Split}_{\mathbb{S}}(s)$ *is a randomized algorithm that takes in a secret s, and outputs n shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$.*
- $\mathsf{Reconstruct}_{\mathbb{S}}(\{\mathsf{sh}_i\}_{i\in P})$ *is a deterministic algorithm that takes in a set of shares $\{\mathsf{sh}_i\}_{i\in P}$ for some $P \subseteq [n]$, and outputs either s or $\bot$.*

The scheme should satisfy the following notions of correctness and security.

- ***Correctness.*** *For all subsets $P \subseteq [n]$ such that there exists $S \in \mathbb{S}$ such that $P \subseteq S$,*

$$\Pr\left[\mathsf{Reconstruct}(\{\mathsf{sh}_i\}_{i\in P}) = s : (\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{Split}_{\mathbb{S}}(s)\right] = 1.$$

- ***Privacy.*** *There exists a randomized algorithm* Sim *such that for all subsets $P \subseteq [n]$ such that for all $S \in \mathbb{S}$, $P \not\subseteq S$, and any s,*

$$\{\{\mathsf{sh}_i\}_{i\in P} : (\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{Split}_{\mathbb{S}}(s)\} \equiv \{\{\mathsf{sh}_i\}_{i\in P} : \{\mathsf{sh}_i\}_{i\in P} \leftarrow \mathsf{Sim}(P)\}\,.$$

2-out-of-2 secret sharing with certified deletion Now, we recall the definition of 2-out-of-2 secret sharing *with certified deletion* as defined by [BK23]. A 2-out-of-2 secret sharing scheme is a very special case of secret sharing where the secret is split into two shares such that both shares together determine the secret, but either share individually cannot be used to recover the secret.

Definition 12.4.2 (2-out-of-2 Secret Sharing with Certified Deletion)**.** *We augment the standard notion of secret sharing to include a deletion algorithm* $\mathsf{Delete}_{2\text{-}2}$ *and a verification algorithm* $\mathsf{Verify}_{2\text{-}2}$*. We also specify that one share is quantum and the other share is classical. Finally, we introduce a security parameter 1^λ, since our deletion security guarantee will be statistical rather than perfect.*

- $\mathsf{Split}_{2\text{-}2}(1^\lambda, s)$ *is a quantum algorithm that takes in the security parameter 1^λ, a secret s, and outputs a quantum share $|\mathsf{sh}_1\rangle$, a classical share sh_2, and a classical verification key* vk*.*
- $\mathsf{Reconstruct}_{2\text{-}2}(|\mathsf{sh}_1\rangle, \mathsf{sh}_2)$ *is a quantum algorithm that takes in two shares and outputs the secret s.*
- $\mathsf{Delete}_{2\text{-}2}(|\mathsf{sh}_1\rangle)$ *is a quantum algorithm that takes in a quantum share $|\mathsf{sh}_1\rangle$ and outputs a deletion certificate* cert*.*
- $\mathsf{Verify}_{2\text{-}2}(\mathsf{vk}, \mathsf{cert})$ *is a classical algorithm that takes in a verification key* vk *and a deletion certificate* cert *and outputs either $\top$ or $\bot$.*

Beyond satisfying the standard secret sharing notions of correctness and privacy (definition 12.4.1) for the 2-out-of-2 access structure, the scheme should satisfy the following properties.

- ***Deletion correctness.*** *For all $\lambda \in \mathbb{N}$ and s,*

$$\Pr\left[\mathsf{Verify}_{2\text{-}2}(\mathsf{vk}, \mathsf{cert}) = \top : \begin{array}{r}(|\mathsf{sh}_1\rangle, \mathsf{sh}_2) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\lambda, s)\\ \mathsf{cert} \leftarrow \mathsf{Delete}_{2\text{-}2}(|\mathsf{sh}_1\rangle)\end{array}\right] = 1.$$

- ***Certified deletion security.*** *Let* Adv *be an adversary, s be a secret, and define the experiment* $2\mathsf{SS\text{-}NSCD}(1^\lambda, \mathsf{Adv}, s)$ *as follows:*

- *Sample* $(|\mathsf{sh}_1\rangle, \mathsf{sh}_2, \mathsf{vk}) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\lambda, s)$.
- *Run* $(\mathsf{cert}, \mathcal{R}) \leftarrow \mathsf{Adv}(1^\lambda, |\mathsf{sh}_1\rangle)$*, where* $\mathcal{R}$ *is an arbitrary output register.*
- *If* $\mathsf{Verify}_{2\text{-}2}(\mathsf{vk}, \mathsf{cert}) = \top$*, output* $(\mathcal{R}, \mathsf{sh}_2)$*, and otherwise output* $\bot$.

Then, for any unbounded adversary Adv *and any pair of secrets* s_0, s_1*, it holds that*

$$\mathsf{TD}\left[2\mathsf{SS\text{-}NSCD}(1^\lambda, \mathsf{Adv}, s_0), 2\mathsf{SS\text{-}NSCD}(1^\lambda, \mathsf{Adv}, s_1)\right] = 2^{-\Omega(\lambda)}.$$

[BK23] showed the existence of a 2-out-of-2 secret sharing scheme with certified deletion satisfying the above definition. Notice that our certified deletion security definition requires a trace distance of $2^{-\Omega(\lambda)}$. While the theorem from [BK23] only states a bound of $\mathsf{negl}(\lambda)$, a quick inspection of their proof establishes that they in fact show a bound of $2^{-\Omega(\lambda)}$.

Chapter 13

High-Rate Seedless Quantum Extractors

In this section, we study seedless extraction of large amounts of entropy from a quantum source. The source of entropy comes from applying a quantum Fourier transform to a state which is "almost" a computational basis state. In particular, the source register $\mathcal{X}$ is in superposition over vectors with low Hamming weight, and may be arbitrarily entangled with a register $\mathcal{A}$ that contains side-information about the source. Previously, [ABKK23] showed that the XOR function perfectly extracts a single bit of entropy in this setting. However, in order to extract multiple bits of entropy, they resorted to the use of a random oracle. We also remark that the case of *seeded* extraction has been well-studied by, e.g. [RK05, DFL+09, BF10].

We describe a general class of extractors that produces multiple truly random elements of any finite field $\mathbb{F}$, even conditioned on the side-information register $\mathcal{A}$. In the case where the finite field has order p^k for a prime p, we show that a large amount of entropy is generated even when the quantum Fourier transform is applied by interpreting each element $x \in \mathbb{F}_{p^k}$ as a vector $\mathbf{x} \in \mathbb{F}_p^k$ and applying the transform mod p to each index (as opposed to applying the transform mod p^k directly to the field element). This feature allows the source to be prepared using less entanglement in our eventual application to secret sharing.

Notation. The Hamming weight $h_{\mathbb{F}}(\mathbf{v})$ of a vector $\mathbf{v} \in \mathbb{F}^M$ over a finite field $\mathbb{F}$ is its number of non-zero positions. We denote vectors $\mathbf{v}$ and matrices $\mathbf{R}$ using bold font. Since we will be working with elements which can be interpreted as elements of two different fields, we use $(\cdot)_{\mathbb{F}}$ to denote that the contents of the parentheses should be interpreted as elements and operations over $\mathbb{F}$. For example, $(x+y)_{\mathbb{F}}$ denotes addition of x and y inside the field $\mathbb{F}$. If $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$, then $(\mathbf{x}+\mathbf{y})_{\mathbb{F}}$ denotes vector addition. For an extension field $\mathbb{K}$ of $\mathbb{F}$, a scalar $x \in \mathbb{K}$ can also be interpreted as a vector $\mathbf{x} \in \mathbb{F}^k$. In this case, for $x, y \in \mathbb{K}$, $(\mathbf{x} \cdot \mathbf{y})_{\mathbb{F}}$ produces a scalar in $\mathbb{F}$. If an element can be interpreted as either a vector or a scalar, we bold it depending on the context of the first operation applied; for example, $(x\mathbf{y})_{\mathbb{K}}$ or $(\mathbf{x} \cdot \mathbf{z})_{\mathbb{F}}$ for $x \in \mathbb{K}$, $\mathbf{y} \in \mathbb{K}^n$, and $\mathbf{z} \in \mathbb{F}^k$.

Theorem 13.0.1. *Let $\mathbb{F}$ be a finite field of order p^k. Let $X = \mathcal{X}_1, \ldots, \mathcal{X}_M$ be a register containing M $\mathbb{F}$-qudits, and consider any quantum state*

$$|\gamma\rangle^{\mathcal{A},\mathcal{X}} = \sum_{\mathbf{u} \in \mathbb{F}^M : h_{\mathbb{F}}(\mathbf{u}) < \frac{M-m}{2}} |\psi_{\mathbf{u}}\rangle^{\mathcal{A}} \otimes |\mathbf{u} + \mathbf{w}\rangle^{\mathcal{X}}$$

for some integer $m \leq M$ and some fixed string $\mathbf{w} \in \mathbb{F}^M$. Let $\mathbf{R} \in \mathbb{F}^{m \times M}$ be a matrix such that every set of m columns of $\mathbf{R}$ are linearly independent.

Let $\rho^{\mathcal{A},\mathcal{Y}}$ be the mixed state that results from the following procedure:

1. *Apply a quantum Fourier transform over $\mathbb{F}$'s additive group $\mathbb{Z}_p^k$ to each register $\mathcal{X}_i$. In other words, interpret $\mathcal{X}_i$ as a sequence of registers $\mathcal{X}_{i,1}, \ldots, \mathcal{X}_{i,k}$ containing $\mathbb{Z}_p$-qudits, then apply a quantum Fourier transform mod p to each $\mathcal{X}_{i,j}$.*

2. *Initialize a fresh register $\mathcal{Y}$, and apply the isometry $|\mathbf{x}\rangle^{\mathcal{X}} \mapsto |\mathbf{x}\rangle^{\mathcal{X}} \otimes |\mathbf{R}\mathbf{x}\rangle^{\mathcal{Y}}$.*

3. *Trace out register $\mathcal{X}$.*

Then

$$\rho^{\mathcal{A},\mathcal{Y}} = \mathsf{Tr}^X\left[|\gamma\rangle\langle\gamma|\right] \otimes \left(\frac{1}{|\mathbb{F}|^m} \sum_{\mathbf{y}\in\mathbb{F}^m} |\mathbf{y}\rangle\langle\mathbf{y}|\right).$$

Remark 13.0.2. *As an example, consider $\mathbb{F}$ to be the field with 2^n elements. Note that for the source, the Hamming weight is taken over the larger field $\mathbb{F}$, but the quantum Fourier transform is done over the individual qubits, which in this case makes it just a Hadamard gate. The extractor R operates over $\mathbb{F}$ and produces an output in $\mathbb{F}^m$.*

Proof. First, we apply the Fourier transform to $|\gamma\rangle$ to obtain

$$\sqrt{|\mathbb{F}|^{-M}} \sum_{\mathbf{u}\in\mathbb{F}^M : h_{\mathbb{F}}(\mathbf{u}) < \frac{M-m}{2}} |\psi_{\mathbf{u}}\rangle^{\mathcal{A}} \otimes \sum_{\mathbf{x}\in\mathbb{F}^M} \omega_p^{((\mathbf{u}+\mathbf{w})_{\mathbb{F}}\cdot\mathbf{x})_{\mathbb{Z}_p}} |\mathbf{x}\rangle^{\mathcal{X}},$$

where ω_p is a primitive p'th root of unity. Next, after applying the extractor, but before tracing out $\mathcal{X}$, the state becomes

$$\sqrt{|\mathbb{F}|^{-M}} \sum_{\mathbf{x}\in\mathbb{F}^M} \left(\sum_{\mathbf{u}\in\mathbb{F}^M : h_{\mathbb{F}}(\mathbf{u}) < \frac{M-m}{2}} \omega_p^{((\mathbf{u}+\mathbf{w})_{\mathbb{F}}\cdot\mathbf{x})_{\mathbb{Z}_p}} |\psi_{\mathbf{u}}\rangle^{\mathcal{A}} \right) \otimes |\mathbf{x}\rangle^{\mathcal{X}} \otimes |\mathbf{R}\mathbf{x}\rangle^{\mathcal{Y}} \tag{13.1}$$

$$\coloneqq \sqrt{|\mathbb{F}|^{-M}} \sum_{\mathbf{x}\in\mathbb{F}^M} |\phi_{\mathbf{x}}\rangle^{\mathcal{A}} \otimes |\mathbf{x}\rangle^{\mathcal{X}} \otimes |\mathbf{R}\mathbf{x}\rangle^{\mathcal{Y}}. \tag{13.2}$$

Since the additive group of $\mathbb{F}$ is $\mathbb{Z}_p^k$, for every $\mathbf{u}, \mathbf{w} \in \mathbb{F}^M$ and $\mathbf{x} \in \mathbb{Z}_p^{kM}$, we have

$$((\mathbf{u}+\mathbf{w})_{\mathbb{F}}\cdot\mathbf{x})_{\mathbb{Z}_p} = ((\mathbf{u}+\mathbf{w})_{\mathbb{Z}_p^k}\cdot\mathbf{x})_{\mathbb{Z}_p} = ((\mathbf{u}+\mathbf{w})\cdot\mathbf{x})_{\mathbb{Z}_p}.$$

Tracing out register $\mathcal{X}$ yields

$$\rho^{\mathcal{A},\mathcal{Y}} = |\mathbb{F}|^{-M} \sum_{\mathbf{x}\in\mathbb{F}^M} |\phi_{\mathbf{x}}\rangle\langle\phi_{\mathbf{x}}| \otimes |\mathbf{R}\mathbf{x}\rangle\langle\mathbf{R}\mathbf{x}| \tag{13.3}$$

$$= |\mathbb{F}|^{-M} \sum_{\substack{\mathbf{y}\in\mathbb{F}^m \\ \mathbf{x}\in\mathbb{F}^M:(\mathbf{R}\mathbf{x})_{\mathbb{F}}=\mathbf{y}}} |\phi_{\mathbf{x}}\rangle\langle\phi_{\mathbf{x}}| \otimes |\mathbf{y}\rangle\langle\mathbf{y}| \tag{13.4}$$

$$= |\mathbb{F}|^{-M} \sum_{\substack{\mathbf{y}\in\mathbb{F}^m \\ \mathbf{x}\in\mathbb{F}^M:(\mathbf{R}\mathbf{x})_{\mathbb{F}}=\mathbf{y}}} \left(\sum_{\substack{\mathbf{u}_1,\mathbf{u}_2\in\mathbb{F}^M: \\ h_{\mathbb{F}}(\mathbf{u}_1),h_{\mathbb{F}}(\mathbf{u}_2)\leq\frac{M-m}{2}}} \omega_p^{((\mathbf{u}_1+\mathbf{w})\cdot\mathbf{x})_{\mathbb{Z}_p}} \overline{\omega_p^{((\mathbf{u}_2+\mathbf{w})\cdot\mathbf{x})_{\mathbb{Z}_p}}} |\phi_{\mathbf{u}_1}\rangle\langle\phi_{\mathbf{u}_2}| \right) \otimes |\mathbf{y}\rangle\langle\mathbf{y}| \tag{13.5}$$

$$= \sum_{\substack{\mathbf{u}_1,\mathbf{u}_2\in\mathbb{F}^M: \\ h_{\mathbb{F}}(\mathbf{u}_1),h_{\mathbb{F}}(\mathbf{u}_2)\leq\frac{M-m}{2}}} |\phi_{\mathbf{u}_1}\rangle\langle\phi_{\mathbf{u}_2}| \otimes \left(|\mathbb{F}|^{-M} \sum_{\mathbf{y}\in\mathbb{F}^m} |\mathbf{y}\rangle\langle\mathbf{y}| \sum_{\mathbf{x}\in\mathbb{F}^M:(\mathbf{R}\mathbf{x})_{\mathbb{F}}=\mathbf{y}} \omega_p^{((\mathbf{u}_1+\mathbf{w})\cdot\mathbf{x}-(\mathbf{u}_2+\mathbf{w})\cdot\mathbf{x})_{\mathbb{Z}_p}} \right) \tag{13.6}$$

$$= \sum_{\substack{\mathbf{u}_1,\mathbf{u}_2\in\mathbb{F}^M: \\ h_{\mathbb{F}}(\mathbf{u}_1),h_{\mathbb{F}}(\mathbf{u}_2)\leq\frac{M-m}{2}}} |\phi_{\mathbf{u}_1}\rangle\langle\phi_{\mathbf{u}_2}| \otimes \left(|\mathbb{F}|^{-M} \sum_{\mathbf{y}\in\mathbb{F}^m} |\mathbf{y}\rangle\langle\mathbf{y}| \sum_{\mathbf{x}\in\mathbb{F}^M:(\mathbf{R}\mathbf{x})_{\mathbb{F}}=\mathbf{y}} \omega_p^{((\mathbf{u}_1-\mathbf{u}_2)\cdot\mathbf{x})_{\mathbb{Z}_p}} \right). \tag{13.7}$$

Next, we apply Claim 13.0.3, proven below, to show that every $|\phi_{\mathbf{u}_1}\rangle\langle\phi_{\mathbf{u}_2}|$ term where $\mathbf{u}_1 \neq \mathbf{u}_2$ has coefficient 0. To see this, consider any such $\mathbf{u}_1, \mathbf{u}_2$ and the value $\mathbf{u} = (\mathbf{u}_1-\mathbf{u}_2)_{\mathbb{Z}_p} = (\mathbf{u}_1-\mathbf{u}_2)_{\mathbb{F}}$. Condition 1 is satisfied by $\mathbf{u}$ since $\mathbf{u}_1 \neq \mathbf{u}_2$. Condition 2 is satisfied since $h_{\mathbb{F}}(\mathbf{u}) \leq h_{\mathbb{F}}(\mathbf{u}_1) + h_{\mathbb{F}}(\mathbf{u}_2) \leq M - m$, so there are at least m indices of $\mathbf{u}$ which are zero. Finally, condition 3 is satisfied since any m columns of R are linearly independent.

Finally, noting that if $\mathbf{u}_1 = \mathbf{u}_2$, then the coefficient of $|\phi_{\mathbf{u}_1}\rangle\langle\phi_{\mathbf{u}_1}| \otimes |\mathbf{y}\rangle\langle\mathbf{y}|$ is the number of solutions to $(\mathbf{R}\mathbf{x})_{\mathbb{F}} = \mathbf{y}$, which is $|\mathbb{F}|^{M-m}$, we conclude that

$$\rho^{\mathcal{A},\mathcal{Y}} = \sum_{\mathbf{u}\in\mathbb{F}^M:h_{\mathbb{F}}(\mathbf{u})\leq\frac{M-m}{2}} |\phi_{\mathbf{u}}\rangle\langle\phi_{\mathbf{u}}| \otimes \left(|\mathbb{F}|^{-m} \sum_{\mathbf{y}\in\mathbb{F}^m} |\mathbf{y}\rangle\langle\mathbf{y}| \right) \tag{13.8}$$

$$= \mathsf{Tr}^X[|\gamma\rangle\langle\gamma|] \otimes \left(|\mathbb{F}|^{-m} \sum_{\mathbf{y}\in\mathbb{F}^m} |\mathbf{y}\rangle\langle\mathbf{y}| \right). \tag{13.9}$$

□

Claim 13.0.3. *Let $\mathbf{u} \in \mathbb{F}^M$ and $\mathbf{y} \in \mathbb{F}^m$, and suppose that*

1. *$\mathbf{u}_i \neq 0$ for some index i.*

2. *There exists a set $J \subseteq [0, \ldots, M-1]$ of size m such that for every $j \in J$, $\mathbf{u}_j = 0$.*

3. *The submatrix $\mathbf{R}_J$ consisting of the columns of $\mathbf{R}$ corresponding to J has full rank.*

Then

$$\sum_{\mathbf{x} \in \mathbb{F}^M : (\mathbf{R}\mathbf{x})_{\mathbb{F}} = \mathbf{y}} \omega_p^{(\mathbf{u} \cdot \mathbf{x})_{\mathbb{Z}_p}} = 0.$$

Remark 13.0.4. *We note that in the case that $\mathbb{F} = \mathbb{F}_p$, then the above expression actually holds for any $\mathbf{u} \notin \mathsf{rowspan}(\mathbf{R})$, which follows from a standard argument. The three conditions above do imply that $\mathbf{u} \notin \mathsf{rowspan}(\mathbf{R})$, but are more restrictive. We take advantage of the extra restrictions in order to prove that the expression holds even in the case where $\mathbb{F}$ is an extension field of $\mathbb{F}_p$.*

Proof. Our strategy will be to partition the affine subspace $S_{\mathbf{y}} = \{\mathbf{x} \in \mathbb{F}^M : (\mathbf{R}\mathbf{x})_{\mathbb{F}} = \mathbf{y}\}$ into parallel lines, and then claim that the sum over each line is 0.

To begin, define a vector $\mathbf{z} \in \mathbb{F}^M$ so that

- $\mathbf{z}_i = 1$,
- $\mathbf{z}_j = 0$ for all $j \notin J \cup \{i\}$, and
- $(\mathbf{R}\mathbf{z})_{\mathbb{F}} = 0^m$,

which is possible because the $m \times m$ submatrix $\mathbf{R}_J$ has full rank. By construction, we have that for any $c \in \mathbb{F}$,

$$(\mathbf{u} \cdot (c\mathbf{z})_{\mathbb{F}})_{\mathbb{Z}_p} = \left(\mathbf{u}_i \cdot (c\mathbf{z}_i)_{\mathbb{F}} + \sum_{j \in J} \mathbf{u}_j \cdot (c\mathbf{z}_j)_{\mathbb{F}} + \sum_{j \notin J \cup \{i\}} \mathbf{u}_j \cdot (0)_{\mathbb{F}} \right)_{\mathbb{Z}_p} = (\mathbf{u}_i \cdot \mathbf{c})_{\mathbb{Z}_p}, \tag{13.10}$$

where note that in the final expression, $\mathbf{u}_i$ and $\mathbf{c}$ are interpreted as vectors in $\mathbb{Z}_p^k$.

Now, fix any $\mathbf{x} \in S_{\mathbf{y}}$ and $c \in \mathbb{F}$. Then we have that $(\mathbf{x} + c\mathbf{z})_{\mathbb{F}} \in S_{\mathbf{y}}$, since $(\mathbf{R}(\mathbf{x} + c\mathbf{z}))_{\mathbb{F}} = \mathbf{y} + 0$. Therefore, we can partition $S_{\mathbf{y}}$ into one-dimensional cosets (lines) of the form $\{\mathbf{x} + c\mathbf{z}\}_{c \in \mathbb{F}}$.

We now show that the sum over any particular coset is 0, i.e. that for any $\mathbf{x} \in S_{\mathbf{y}}$,

$$\sum_{c \in \mathbb{F}} \omega_p^{(\mathbf{u} \cdot (\mathbf{x} + c\mathbf{z})_{\mathbb{F}})_{\mathbb{Z}_p}} = 0.$$

Since the additive group of $\mathbb{F}$ is $\mathbb{Z}_p^k$, by eq. (13.10) we have that

$$\begin{aligned}(\mathbf{u} \cdot (\mathbf{x} + c\mathbf{z})_{\mathbb{F}})_{\mathbb{Z}_p} &= (\mathbf{u} \cdot \mathbf{x} + \mathbf{u} \cdot (c\mathbf{z})_{\mathbb{F}})_{\mathbb{Z}_p} \\ &= (\mathbf{u} \cdot \mathbf{x} + \mathbf{u}_i \cdot \mathbf{c})_{\mathbb{Z}_p}\end{aligned}$$

We now view $\mathbf{u}_i \in \mathbb{F}$ as a vector $\mathbf{u}_i = u_{i,0}, \ \ldots, \ u_{i,k-1} \in \mathbb{Z}_p^k$. In particular, since $\mathbf{u}_i \neq 0$, there exists an index t such that $u_{k,t} \neq 0 \in \mathbb{Z}_p$. By also interpreting $\mathbf{c} \in \mathbb{F}$ as an element of $\mathbb{Z}_p^k$, we can decompose it as $\mathbf{c} = (\mathbf{c}' + c_t \mathbf{e}_t)_{\mathbb{Z}_p}$, where $\mathbf{c}' \in \mathbb{Z}_p^k$ is such that $\mathbf{c}'_t = 0$, where $c_t \in \mathbb{Z}_p$, and where $\mathbf{e_t} \in \mathbb{Z}_p^k$ is the t'th standard basis vector. Therefore

$$(\mathbf{u} \cdot (\mathbf{x} + c\mathbf{z})_{\mathbb{F}})_{\mathbb{Z}_p} = (\mathbf{u} \cdot \mathbf{x} + \mathbf{u}_i \cdot \mathbf{c}' + u_{i,t} c_t)_{\mathbb{Z}_p}.$$

Since $u_{i,t} \neq 0$ and ω_p is a primitive p'th root of unity, we know that $\omega_p^{u_{i,t}} \neq 1$ is a p'th root of unity. Therefore by Claim 12.1.1,

$$\sum_{c \in \mathbb{F}} \omega_p^{(\mathbf{u} \cdot (\mathbf{x} + \mathbf{cz})_{\mathbb{F}})_{\mathbb{Z}_p}} = \sum_{\mathbf{c}' \in \mathbb{Z}_p^k : \mathbf{c}'_t = 0} \omega_p^{(\mathbf{u} \cdot \mathbf{x} + \mathbf{u}_i \cdot \mathbf{c}')_{\mathbb{Z}_p}} \cdot \sum_{c_t \in \mathbb{Z}_p} \left(\omega_p^{u_{i,t}}\right)^{c_t} \tag{13.11}$$

$$= \sum_{\mathbf{c}' \in \mathbb{Z}_p^k : \mathbf{c}'_t = 0} \omega_p^{(\mathbf{u} \cdot \mathbf{x} + \mathbf{u}_i \cdot \mathbf{c}')_{\mathbb{Z}_p}} \cdot 0 \tag{13.12}$$

$$= 0. \tag{13.13}$$

□

Chapter 14

Definitions of Secret Sharing with Certified Deletion

A secret sharing scheme with certified deletion augments the syntax of a secret sharing scheme with additional algorithms to delete shares and verify deletion certificates. We define it for general access structures. As described in section 12.4, an access structure $\mathbb{S} \subseteq \mathcal{P}([n])$ for n parties is a monotonic set of sets, i.e. if $S \in \mathbb{S}$ and $S' \supset S$, then $S' \in \mathbb{S}$. Any set of parties $S \in \mathbb{S}$ is authorized to access the secret. A simple example of an access structure is the threshold structure, where any set of at least k parties is authorized to access the secret. We denote this access structure as (k, n).

Definition 14.0.1 (Secret Sharing with Certified Deletion)**.** *A secret sharing scheme with certified deletion is specified by a monotone access structure $\mathbb{S}$ over n parties, and consists of four algorithms:*

- $\mathsf{Split}_{\mathbb{S}}(1^\lambda, s)$ *takes in a secret s, and outputs n share registers $\mathcal{S}_1, \ldots, \mathcal{S}_n$ and a verification key* vk.
- $\mathsf{Reconstruct}_{\mathbb{S}}(\{S_i\}_{i\in P})$ *takes in set of share registers for some $P \subseteq [n]$, and outputs either s or $\perp$.*
- $\mathsf{Delete}_{\mathbb{S}}(\mathcal{S}_i)$ *takes in a share register and outputs a certificate of deletion* cert.
- $\mathsf{Verify}_{\mathbb{S}}(\mathsf{vk}, i, \mathsf{cert})$ *takes in the verification key* vk*, an index i, and a certificate of deletion* cert*, and outputs either $\top$ (indicating accept) or $\perp$ (indicating reject).*

Definition 14.0.2 (Correctness of Secret Sharing with Certified Deletion)**.** *A secret sharing scheme with certified deletion must satisfy two correctness properties:*

- ***Reconstruction Correctness.*** *For all $\lambda \in \mathbb{N}$ and all sets $S \in \mathbb{S}$,*

$$\Pr\left[\mathsf{Reconstruct}_{\mathbb{S}}(\{\mathcal{S}_i\}_{i\in S}) : (\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathsf{vk}) \leftarrow \mathsf{Split}_{\mathbb{S}}(1^\lambda, s)\right] = 1.$$

- ***Deletion Correctness.*** *For all $\lambda \in \mathbb{N}$ and all $i \in [n]$,*

$$\Pr\left[\mathsf{Verify}_{\mathbb{S}}(\mathsf{vk}, i, \mathsf{cert}) = \top : \begin{array}{r}(\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathsf{vk}) \leftarrow \mathsf{Split}_{\mathbb{S}}(1^\lambda, s)\\ \mathsf{cert} \leftarrow \mathsf{Delete}_{\mathbb{S}}(\mathcal{S}_i)\end{array}\right] = 1.$$

The standard notion of security for secret sharing requires that no set of unauthorized shares $S \notin \mathbb{S}$ reveals any information about the secret (see section 12.4). We next present our notion of *no-signaling certified deletion security*. Here, the shares are partitioned into unauthorized sets, and different parts of the

adversary operate on each partition, potentially deleting some number of shares from each. The different parts of the adversary are allowed to share entanglement, but are not allowed to signal. If the adversaries jointly produce a valid certificate for at least one share from every authorized set, then we require that the joint residual state of *all* of the adversaries contains no (or negligible) information about the secret. Observe that this notion of security is at least as strong as the standard notion of security for secret sharing (if we relax to statistical rather than perfect security). Indeed, if the standard notion does not hold, and thus there is some unauthorized set S that leaks information about the secret, then the adversary would be able to win the certified deletion game by honestly deleting every share except for those in S.

Definition 14.0.3 (No-Signaling Certified Deletion Security for Secret Sharing)**.** *Let $P = (P_1, \ldots, P_\ell)$ be a partition of $[n]$, let $|\psi\rangle$ be an ℓ-part state on registers $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$, and let $\mathsf{Adv} = (\mathsf{Adv}_1, \ldots, \mathsf{Adv}_\ell)$ be an ℓ-part adversary. Define the experiment $\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s)$ as follows:*

1. *Sample $(\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathsf{vk}) \leftarrow \mathsf{Split}_{\mathbb{S}}(1^\lambda, s)$.*
2. *For each $t \in [\ell]$, run $(\{\mathsf{cert}_i\}_{i \in P_t}, \mathcal{R}'_t) \leftarrow \mathsf{Adv}_t(\{\mathcal{S}_i\}_{i \in P_t}, \mathcal{R}_t)$, where $\mathcal{R}'_t$ is an arbitrary output register.*
3. *If for all $S \in \mathbb{S}$, there exists $i \in S$ such that $\mathsf{Verify}_{\mathbb{S}}(\mathsf{vk}, i, \mathsf{cert}_i) = \top$, then output $(\mathcal{R}'_1, \ldots, \mathcal{R}'_\ell)$, and otherwise output $\bot$.*

A secret sharing scheme for access structure $\mathbb{S}$ has no-signaling certified deletion security if for any "admissible" partition $P = (P_1, \ldots, P_\ell)$ (i.e. for all $P_t \in P$ and $S \in \mathbb{S}$, $P_t \not\subseteq S$), any ℓ-part state $|\psi\rangle$, any (unbounded) ℓ-part adversary Adv, and any pair of secrets s_0, s_1,

$$\mathsf{TD}[\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s_0),\ \mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s_1)] = \mathsf{negl}(\lambda).$$

Next, we present an alternative definition which allows the adversary to start by corrupting some unauthorized set, and then continue to adaptively delete some shares and corrupt new parties, as long as the total set of parties corrupted minus the set of shares deleted is unauthorized. Similarly to the previous definition, adaptive certified deletion for secret sharing subsumes the standard notion of security for secret sharing.

Definition 14.0.4 (Adaptive Certified Deletion for Secret Sharing)**.** *Let Adv be an adversary with internal register $\mathcal{R}$ which is initialized to a state $|\psi\rangle$, let $\mathbb{S}$ be an access structure, and let s be a secret. Define the experiment $\mathsf{SS\text{-}ACD}_{\mathbb{S}}(1^\lambda, |\psi\rangle, \mathsf{Adv}, s)$ as follows:*

1. *Sample $(\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathsf{vk}) \leftarrow \mathsf{Split}_{\mathbb{S}}(1^\lambda, s)$. Initialize the corruption set $C = \emptyset$ and the deleted set $D = \emptyset$.*
2. *In each round i, the adversary may do one of three things:*
   (a) *End the experiment by outputting a register $\mathcal{R} \leftarrow \mathsf{Adv}(\{\mathcal{S}_j\}_{j \in C}, \mathcal{R})$.*
   (b) *Delete a share by outputting a certificate cert_i, an index $j_i \in [n]$, and register $(\mathsf{cert}_i, j_i, \mathcal{R}) \leftarrow \mathsf{Adv}(\{\mathcal{S}_j\}_{j \in C}, \mathcal{R})$. When the adversary chooses this option, if $\mathsf{Verify}_{\mathbb{S}}(\mathsf{vk}, j_i, \mathsf{cert}_i)$ outputs $\top$, then add j_i to D. Otherwise, immediately abort the experiment and output $\bot$.*
   (c) *Corrupt a new share by outputting an index $j_i \in [n]$ and register $(j_i, \mathcal{R}) \leftarrow \mathsf{Adv}(\{\mathcal{S}_j\}_{j \in C}, \mathcal{R})$. When the adversary chooses this option, add j_i to C. If $C \backslash D \in \mathbb{S}$, immediately abort the experiment and output $\bot$.*
3. *Output $\mathcal{R}$, unless the experiment has already aborted.*

A secret sharing scheme for access structure $\mathbb{S}$ has adaptive certified deletion security if for any (unbounded) adversary Adv*, any state* $|\psi\rangle$*, and any pair of secrets* (s_0, s_1)*,*

$$\mathsf{TD}[\mathsf{SS\text{-}ACD}_{\mathbb{S}}(1^\lambda, |\psi\rangle, \mathsf{Adv}, s_0),\ \mathsf{SS\text{-}ACD}_{\mathbb{S}}(1^\lambda, |\psi\rangle, \mathsf{Adv}, s_1)] = \mathsf{negl}(\lambda)$$

It will also be convenient to establish some notation for the order of the corrupted and deleted shares. Let c_a be the a'th share to be corrupted (i.e. added to C) and let d_b be the b'th share to be deleted (i.e. added to D).

Chapter 15

Secret Sharing with No-Signaling Certified Deletion

In this section, we'll show how to combine any classical secret sharing scheme $(\mathsf{CSplit}_{\mathbb{S}}, \mathsf{CReconstruct}_{\mathbb{S}})$ (definition 12.4.1) for access structure $\mathbb{S} \in \mathcal{P}([n])$ with a 2-out-of-2 secret sharing scheme with certified deletion $(\mathsf{Split}_{2\text{-}2}, \mathsf{Reconstruct}_{2\text{-}2}, \mathsf{Delete}_{2\text{-}2}, \mathsf{Verify}_{2\text{-}2})$ (definition 12.4.2) in order to obtain a secret sharing scheme for $\mathbb{S}$ that satisfies no-signaling certified deletion security (definition 14.0.3).

Theorem 15.0.1. *The construction given in fig. 15.1 satisfies no-signaling certified deletion security (definition 14.0.3).*

Proof. Let $\mathsf{Adv} = (\mathsf{Adv}_1, \ldots, \mathsf{Adv}_\ell)$ be any ℓ-part adversary that partitions the shares using an admissible partition $P = (P_1, \ldots, P_\ell)$ and is initialized with the ℓ-part state $|\psi\rangle$ on registers $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$. Let s_0, s_1 be any two secrets, and assume for contradiction that

$$\mathsf{TD}[\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s_0),\ \mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s_1)] = \mathsf{nonnegl}(\lambda).$$

Now, for $s \in \{s_0, s_1\}$, define a hybrid $\mathcal{H}_1(s)$ as follows.

$\underline{\mathcal{H}_1(s)}$

1. Sample $C \leftarrow \mathcal{P}([n])$.
2. Sample $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(s)$.
3. Set $\kappa = \max\{\lambda, n\}^2$, and for each $i \in [n]$, sample $(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i, \mathsf{vk}_i) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\kappa, \mathsf{sh}_i)$.
4. For each $i \in [n]$, sample $(\mathsf{csh}_{i,1}, \ldots, \mathsf{csh}_{i,n}) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(\mathsf{csh}_i)$.
5. Set $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, and initialize register $\mathcal{S}_i$ to $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$.
6. For each $t \in [\ell]$, run $(\{\mathsf{cert}_i\}_{i\in P_t}, \mathcal{R}'_t) \leftarrow \mathsf{Adv}_t(\{\mathcal{S}_i\}_{i\in P_t}, \mathcal{R}_t)$.
7. Let $C^* := \{i : \mathsf{Verify}_{2\text{-}2}(\mathsf{vk}_i, i, \mathsf{cert}_i) = \top\}$. Output $(\mathcal{R}'_1, \ldots, \mathcal{R}'_\ell)$ if $C = C^*$ and $[n] \setminus C^* \notin \mathbb{S}$, and otherwise output $\bot$.

That is, $\mathcal{H}_1(s)$ is the same as $\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s)$ except that $\mathcal{H}_1(s)$ makes a uniformly random guess C for the subset of shares for which the adversary produces a valid deletion certificate, and aborts if this guess is incorrect.

$\underline{\mathsf{Split}_{\mathbb{S}}(1^\lambda, s)}$

- Sample $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(s)$.
- Set $\kappa = \max\{\lambda, n\}^2$, and for each $i \in [n]$, sample $(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i, \mathsf{vk}_i) \leftarrow \mathsf{Split}_{\text{2-2}}(1^\kappa, \mathsf{sh}_i)$.
- For each $i \in [n]$, sample $(\mathsf{csh}_{i,1}, \ldots, \mathsf{csh}_{i,n}) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(\mathsf{csh}_i)$.
- Set $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, and initialize register $\mathcal{S}_i$ to $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$.

$\underline{\mathsf{Reconstruct}_{\mathbb{S}}(\{\mathcal{S}_i\}_{i\in P})}$

- Parse each register $\mathcal{S}_i$ as $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$.
- For each $i \in P$, compute $\mathsf{csh}_i \leftarrow \mathsf{CReconstruct}_{\mathbb{S}}(\{\mathsf{csh}_{i,j}\}_{j\in P})$, and output $\bot$ if the result is $\bot$.
- For each $i \in P$, compute $\mathsf{sh}_i \leftarrow \mathsf{Reconstruct}_{\text{2-2}}(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i)$.
- Output $s \leftarrow \mathsf{CReconstruct}_{\mathbb{S}}(\{\mathsf{sh}_i\}_{i\in P})$.

$\underline{\mathsf{Delete}_{\mathbb{S}}(\mathcal{S}_i)}$

- Parse $\mathcal{S}_i$ as $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$ and output $\mathsf{cert} \leftarrow \mathsf{Delete}_{\text{2-2}}(|\mathsf{qsh}_i\rangle)$.

$\underline{\mathsf{Verify}_{\mathbb{S}}(\mathsf{vk}, i, \mathsf{cert})}$

- Parse $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ and output $\mathsf{Verify}_{\text{2-2}}(\mathsf{vk}_i, \mathsf{cert})$.

Figure 15.1: Secret sharing with no-signaling certified deletion security for any access structure $\mathbb{S}$.

Claim 15.0.2. $\mathsf{TD}\left[\mathcal{H}_1(s_0), \mathcal{H}_1(s_1)\right] = \mathsf{nonnegl}(\lambda) \cdot 2^{-n}$.

Proof. This follows directly from the fact that $\mathcal{H}_1(s)$'s guess for C is correct with probability $1/2^n$, and, conditioned on the guess being correct, $\mathcal{H}_1(s)$ is identical to $\mathsf{SS\text{-}NSCD}_{\mathbb{S}}(1^\lambda, P, |\psi\rangle, \mathsf{Adv}, s)$. □

Now, for $s \in \{s_0, s_1\}$ and $k \in [0, \ldots, n]$, define a sequence of hybrids $\mathcal{H}_{2,k}(s)$ as follows.

$\underline{\mathcal{H}_{2,k}(s)}$

1. Sample $C \leftarrow \mathcal{P}([n])$.
2. Sample $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(s)$.
3. Set $\kappa = \max\{\lambda, n\}^2$ and for each $i \in [n]$, if $i \leq k$ and $i \in C$, sample $(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i, \mathsf{vk}_i) \leftarrow \mathsf{Split}_{\text{2-2}}(1^\kappa, \bot)$, and otherwise sample $(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i, \mathsf{vk}_i) \leftarrow \mathsf{Split}_{\text{2-2}}(1^\kappa, \mathsf{sh}_i)$.
4. For each $i \in [n]$, sample $(\mathsf{csh}_{i,1}, \ldots, \mathsf{csh}_{i,n}) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(\mathsf{csh}_i)$.
5. Set $\mathsf{vk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, and initialize register $\mathcal{S}_i$ to $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$.
6. For each $t \in [\ell]$, run $(\{\mathsf{cert}_i\}_{i\in P_t}, \mathcal{R}'_t) \leftarrow \mathsf{Adv}_t(\{\mathcal{S}_i\}_{i\in P_t}, \mathcal{R}_t)$.

7. Let $C^* := \{i : \mathsf{Verify}_{2\text{-}2}(\mathsf{vk}_i, i, \mathsf{cert}_i) = \top\}$. Output $(\mathcal{R}'_1, \ldots, \mathcal{R}'_\ell)$ if $C = C^*$ and $[n] \setminus C^* \notin \mathbb{S}$, and otherwise output $\bot$.

First, note that $\mathcal{H}_1(s_0) = \mathcal{H}_{2,0}(s_0)$ and $\mathcal{H}_1(s_1) = \mathcal{H}_{2,0}(s_1)$. Next, we show the following claim.

Claim 15.0.3. $\mathcal{H}_{2,n}(s_0) \equiv \mathcal{H}_{2,n}(s_1)$.

Proof. In each experiment, if the output is not $\bot$, then we know that $[n] \setminus C$ is an unauthorized set. Moreover, the experiments do not depend on the information $\{\mathsf{sh}_i\}_{i\in C}$. Thus, the claim follows by the perfect privacy of $(\mathsf{CSplit}_{\mathbb{S}}, \mathsf{CReconstruct}_{\mathbb{S}})$, which implies that

$$\begin{aligned}&\{\{\mathsf{sh}_i\}_{i\in C\setminus[n]} : (\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(s_0)\}\\ &\quad\equiv \{\{\mathsf{sh}_i\}_{i\in C\setminus[n]} : (\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(s_1)\}.\end{aligned}$$

□

Finally, we show the following claim.

Claim 15.0.4. *For $s \in \{s_0, s_1\}$ and $k \in [n]$, it holds that* $\mathsf{TD}\left[\mathcal{H}_{2,k-1}(s), \mathcal{H}_{2,k}(s)\right] = 2^{-\Omega(\kappa)}$.

Proof. The only difference between these hybrids is that if $k \in C$, we switch sh_k to $\bot$ in the third step. So, suppose that $k \in C$, and consider the following reduction to the certified deletion security (definition 12.4.2) of $(\mathsf{Split}_{2\text{-}2}, \mathsf{Reconstruct}_{2\text{-}2}, \mathsf{Delete}_{2\text{-}2}, \mathsf{Verify}_{2\text{-}2})$. This experiment is parameterized by a bit b which determines which one of two secrets the certified deletion challenger will share.

- The reduction samples $C \leftarrow \mathcal{P}([n])$ and $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(s)$, and sends $\{\mathsf{sh}_k, \bot\}$ to the challenger.
- The challenger samples $(|\mathsf{qsh}_k\rangle, \mathsf{csh}_k, \mathsf{vk}_k) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\kappa, \mathsf{sh}_k)$ if $b = 0$ or $(|\mathsf{qsh}_k\rangle, \mathsf{csh}_k, \mathsf{vk}_k) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\kappa, \bot)$ if $b = 1$, and sends $|\mathsf{qsh}_k\rangle$ to the reduction.
- For each $i \in [n]\setminus\{k\}$, if $i < k$ and $i \in C$, the reduction samples $(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i, \mathsf{vk}_i) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\kappa, \bot)$, and otherwise samples $(|\mathsf{qsh}_i\rangle, \mathsf{csh}_i, \mathsf{vk}_i) \leftarrow \mathsf{Split}_{2\text{-}2}(1^\kappa, \mathsf{sh}_i)$.
- Let $t^* \in [\ell]$ be such that $k \in P_{t^*}$. For each $i \in [n]\setminus\{k\}$, the reduction samples $(\mathsf{csh}_{i,1}, \ldots, \mathsf{csh}_{i,n}) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(\mathsf{csh}_i)$. Next, the reduction samples $\{\mathsf{csh}_{k,i}\}_{i\in P_{t^*}} \leftarrow \mathsf{Sim}(P_{t^*})$, where Sim is the simulator guaranteed by the privacy of $(\mathsf{CSplit}_{\mathbb{S}}, \mathsf{CReconstruct}_{\mathbb{S}})$.
- For each $i \in P_{t^*}$, initialize register $\mathcal{S}_i$ to $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$.
- The reduction runs $(\{\mathsf{cert}_i\}_{i\in P_{t^*}}, \mathcal{R}'_{t^*}) \leftarrow \mathsf{Adv}_{t^*}(\{\mathcal{S}_i\}_{i\in P_{t^*}}, \mathcal{R}_{t^*})$, and sends cert_k to the challenger.
- The challenger checks whether $\mathsf{Verify}_{2\text{-}2}(\mathsf{vk}_k, \mathsf{cert}_k) = \top$. If so, the challenger returns csh_k, and otherwise the experiment aborts and outputs $\bot$.
- The reduction samples $\{\mathsf{csh}_{k,i}\}_{i\in[n]\setminus P_{t^*}}$ conditioned on the joint distribution of $(\mathsf{csh}_{k,1}, \ldots, \mathsf{csh}_{k,n})$ being identical to $\mathsf{CSplit}_{\mathbb{S}}(\mathsf{csh}_k)$. This is possible due to the guarantee of $\mathsf{Sim}(P_{t^*})$, that is,

$$\begin{aligned}&\left\{\{\mathsf{csh}_{k,i}\}_{i\in P_{t^*}} : (\mathsf{csh}_{k,1}, \ldots, \mathsf{csh}_{k,n}) \leftarrow \mathsf{CSplit}_{\mathbb{S}}(\mathsf{csh}_k)\right\}\\ &\quad\equiv \left\{\{\mathsf{csh}_{k,i}\}_{i\in P_{t^*}} : \{\mathsf{csh}_{k,i}\}_{i\in P_{t^*}} \leftarrow \mathsf{Sim}(P_{t^*})\right\}.\end{aligned}$$

- For each $i \in [n] \setminus P_{t^*}$, the reduction initializes register $\mathcal{S}_i$ to $|\mathsf{qsh}_i\rangle, \{\mathsf{csh}_{j,i}\}_{j\in[n]}$.
- For each $t \in [\ell] \setminus \{t^*\}$, run $(\{\mathsf{cert}_i\}_{i\in P_t}, \mathcal{R}'_t) \leftarrow \mathsf{Adv}_t(\{\mathcal{S}_i\}_{i\in P_t}, \mathcal{R}_t)$.
- Let $C^* := \{i : \mathsf{Verify}_{2\text{-}2}(\mathsf{vk}_i, i, \mathsf{cert}_i) = \top\}$. The reduction outputs $(\mathcal{R}'_1, \ldots, \mathcal{R}'_\ell)$ if $C = C^*$ and $[n] \setminus C^* \notin \mathbb{S}$, and otherwise outputs $\bot$.

Observe that in the case $b = 0$, the output of this experiment is identical to $\mathcal{H}_{2,k-1}(s)$ while if $b = 1$, the output of this experiment is identical to $\mathcal{H}_{2,k}(s)$. Thus, the claim follows from the certified deletion security of $(\mathsf{Split}_{2\text{-}2}, \mathsf{Reconstruct}_{2\text{-}2}, \mathsf{Delete}_{2\text{-}2}, \mathsf{Verify}_{2\text{-}2})$.

□

Thus, by combining claim 15.0.3 and claim 15.0.4, we have that

$$\mathsf{TD}\left[\mathcal{H}_1(s_0), \mathcal{H}_1(s_1)\right] = 2n \cdot 2^{-\Omega(\kappa)} \leq 2^{-\Omega(\{\max\{\lambda,n\}^2\})}.$$

However, this violates claim 15.0.2, since

$$2^{-\Omega(\{\max\{\lambda,n\}^2\})} < \mathsf{nonnegl}(\lambda) \cdot 2^{-n},$$

which completes the proof.

□

Chapter 16

Threshold Secret Sharing with Adaptive Certified Deletion

In this section, we show how to construct a secret sharing scheme for threshold access structures that satisfies adaptive certified deletion (Definition 14.0.4).

16.1 Construction

Our construction is given in Figure 16.2, which uses a set of parameters described in Figure 16.1. We provide some intuition about the parameter settings here.

The secret is encoded in a polynomial f of degree p. For security, we need p to be at least as large as the number of points of f that the adversary can learn. At most, the adversary can hold up to $k-1$ intact shares and the residual states of $n-k+1$ deleted shares. Each of the $k-1$ intact shares contains t' evaluations of f. Additionally, the adversary may retain some small amount of information about each of the deleted shares. We upper bound the retained information by a parameter ℓ for each share. This gives the adversary a maximum of

$$(k-1)t' + (n-k+1)\ell$$

evaluations of f, which becomes the minimum safe setting for p.

Each share will also include a number of "check positions", which contain Fourier basis states that are used for verification of deletion. The number of check positions r and upper bound ℓ are set roughly so that with overwhelming probability, the adversary can retain no more than ℓ evaluations of f when it deletes a share (more precisely, the adversary may retain a *superposition* over potentially different sets of ℓ evaluations). The reader may find it useful to think of ℓ as being the maximum number of unexamined positions in a classical string x when an adversary successfully creates a string y that matches x on r random verification indices. Finally, the total size t of each share is set so that k shares contain less than $(kt-p)/2$ check positions, which is the maximum number of errors that can be corrected in kt evaluations of a polynomial of degree p (see Section 12.3).

Theorem 16.1.1. *There exists secret sharing for threshold access structures which satisfies adaptive certified deletion.*

Proof. The construction is given in Figure 16.2. Deletion correctness is apparent from inspection of the construction. We prove reconstruction correctness in Lemma 16.1.2 and adaptive certified deletion security in Lemma 16.2.1. □

The construction in Figure 16.2 uses the following parameters.[a]

- Each share consists of t total $\mathbb{K}$-registers, where

$$t = (k+1)r\left(1 + \frac{(n-k+1)\log(\lambda)}{\lambda}\right) + 1$$

- A share is divided into r check indices and $t' = t - r$ data indices, where

$$r = (\lambda + (n-k+1)\log(\lambda))^2$$

- ℓ intuitively represents an upper bound on the amount of information which is not destroyed when an adversary generates a valid deletion certificate for a share.

$$\ell = t\frac{\log(\lambda)}{\sqrt{r}}$$

  See the proof of Lemma 16.2.1 for a more precise usage of ℓ.

- The secret will be encoded in a polynomial of degree

$$p = (k-1)t' + (n-k+1)\ell$$

[a]The parameters provided here are slightly looser than necessary, to facilitate easier inspection. We present a tighter set of parameters in Figure 17.1.

Figure 16.1: Parameters for Secret Sharing with Adaptive Certified Deletion

Lemma 16.1.2. *The construction in Figure 16.2 using parameters from Figure 16.1 has reconstruction correctness.*

Proof. The set $\{(it+j, y_{i,j})\}_{i\in P', j\in[t]}$ contains kt pairs which were obtained by measuring k shares. As mentioned in Section 12.3, if all but $e < (kt-p)/2$ of these pairs $(it+j, y_{i,j})$ satisfy $y_{i,j} = f(it+j)$, then $\mathsf{Correct}_{\mathbb{K},p}$ recovers the original polynomial f, where $f(0) = s$. The only points which do not satisfy this are the check positions, of which there are r per share, for a total of kr. Therefore for correctness, we require that

$$\begin{aligned} 2kr &< kt - p && (16.1)\\ &= kt - (k-1)(t-r) - (n-k+1)\ell && (16.2)\\ &= t + (k-1)r - (n-k+1)\ell && (16.3) \end{aligned}$$

Therefore $t - (n-k+1)\ell > (k+1)r$. Substituting $\ell = t\frac{\log(\lambda)}{\sqrt{r}}$ yields

$$t\left(1 - (n-k+1)\frac{\log(\lambda)}{\sqrt{r}}\right) > (k+1)r \quad (16.4)$$

Parameters: Let $\mathbb{F}_2$ be the binary field and let $\mathbb{K}$ be the field with $2^{\lceil \log_2(nt+1) \rceil}$ elements. See Figure 16.1 for descriptions and settings of the parameters t, t', r, ℓ, and p.

$\mathsf{Split}_{(k,n)}(1^\lambda, s)$

- Sample a random polynomial f with coefficients in $\mathbb{K}$ and degree p such that $f(0) = s$.
- For each $i \in [n]$:
  1. Sample a random set of indices $J_i \subset [t]$ of size $t' = t - r$.
  2. For each $j \in J_i$, set $|\psi_{i,j}\rangle = |f(it + j)\rangle$. These are the t' data positions.
  3. For each $j \in [t] \backslash J_i$, sample a uniform element $y_{i,j} \leftarrow \mathbb{K}$ and set $|\psi_{i,j}\rangle = H^{\otimes \lceil \log_2(n+1) \rceil} |y_{i,j}\rangle$. These are the r check positions.
  4. Initialize register $\mathcal{S}_i$ to $\bigotimes_{j=1}^{t} |\psi_{i,j}\rangle$.
- Set $\mathsf{vk} = \{J_i, \{y_{i,j}\}_{j \in [t] \backslash J_i}\}_{i \in [n]}$.

$\mathsf{Reconstruct}_{(k,n)}(1^\lambda, \{\mathcal{S}_i\}_{i \in P})$

- If $|P| < k$, output $\perp$. Otherwise, set P' to be any k shares in P.
- For each $i \in P'$, measure $\mathcal{S}_i$ in the computational basis to obtain $y_i = (y_{i,1}, \ldots, y_{i,t}) \in \mathbb{K}^t$.
- Compute $f \leftarrow \mathsf{Correct}_{\mathbb{K},p}(\{(it + j, y_{i,j})\}_{i \in P', j \in [t]})$, as defined in Section 12.3.
- Output $f(0)$.

$\mathsf{Delete}_{(k,n)}(1^\lambda, \mathcal{S}_i)$

- Parse $\mathcal{S}_i$ as a sequence of $t\lceil \log_2(n+1) \rceil$ single qubit registers, measure each qubit register in Hadamard basis and output the result cert.

$\mathsf{Verify}_{(k,n)}(1^\lambda, \mathsf{vk}, i, \mathsf{cert})$

- Parse $\mathsf{vk} = \{J_i, \{y_{i,j}\}_{j \in [t] \backslash J_i}\}_{i \in [n]}$, and parse $\mathsf{cert} \in \mathbb{K}^t$ as a sequence of t elements of $\mathbb{K}$. Output $\top$ if $\mathsf{cert}_j = y_{i,j}$ for every $j \in J_i$, and $\perp$ otherwise.

Figure 16.2: Construction for Secret Sharing with Adaptive Certified Deletion

$$t > (k+1)r \frac{1}{1 - (n-k+1)\frac{\log(\lambda)}{\sqrt{r}}} \tag{16.5}$$

$$= (k+1)r \frac{\sqrt{r}}{\sqrt{r} - (n-k+1)\log(\lambda)} \tag{16.6}$$

$$= (k+1)r \left(1 + \frac{(n-k+1)\log(\lambda)}{\sqrt{r} - (n-k+1)\log(\lambda)}\right) \tag{16.7}$$

$$= (k+1)r \left(1 + \frac{(n-k+1)\log(\lambda)}{\lambda + (n-k+1)\log(\lambda) - (n-k+1)\log(\lambda)}\right) \tag{16.8}$$

$$= (k+1)r \left(1 + \frac{(n-k+1)\log(\lambda)}{\lambda}\right) \tag{16.9}$$

Note that Equation (16.5) requires that $\left(1-(n-k+1)t\frac{\log(\lambda)}{\sqrt{r}}\right) > 0$. Since the number of check positions is $r = (\lambda + (n-k+1)\log(\lambda))^2$, we have

$$1-(n-k+1)\frac{\log(\lambda)}{\lambda+(n-k+1)\log(\lambda)} > 1-\frac{(n-k+1)\log(\lambda)}{(n-k+1)\log(\lambda)} = 0 \tag{16.10}$$

Finally, observe that the choice of parameters in the construction satisfies these constraints. □

16.2 Proof of Security

Recall that c_a is the a'th share to be corrupted (i.e. added to C) and d_b is the b'th share to be deleted (i.e. added to D). Observe that if c_{k-1+b} is corrupted before d_b is deleted, then $C \backslash D$ has size $\geq k$ and is authorized, so $\mathsf{SS\text{-}ACD}_{(k,n)}$ would abort.

Lemma 16.2.1. *The construction in Figure 16.2 using parameters from Figure 16.1 satisfies adaptive certified deletion for threshold secret sharing.*

We begin by defining a projector which will be useful for reasoning about how many data indices were *not* destroyed when an adversary produces a valid certificate for a share i. A certificate cert_i for share i can be parsed as t elements $\mathsf{cert}_{i,1}, \ldots, \mathsf{cert}_{t,i}$ of $\mathbb{K}$. Denote $\mathsf{cert}'_i = (\mathsf{cert}_{i,j})_{j \in J_i}$ to be the subtuple of elements belonging to data indices. For any certificate cert, we define the projector[1]

$$\Pi_{\mathsf{cert}} = \sum_{\mathbf{u} \in \mathbb{K}^{t'} : h_{\mathbb{K}}(\mathbf{u}) < \ell/2} H^{\otimes t' \lceil \log_2(n+1) \rceil} \left|\mathbf{u} + \mathsf{cert}'\right\rangle \left\langle \mathbf{u} + \mathsf{cert}'\right| H^{\otimes t' \lceil \log_2(n+1) \rceil}$$

Note that H is the Hadamard gate, i.e. it implements a quantum Fourier transform over the binary field $\mathbb{F}_2$, and that the Hamming weight is taken over $\mathbb{K}$.

Let Adv be any adversary which is initialized with a state $|\psi\rangle$ on register $\mathcal{R}$. For $s \in \{s_0, s_1\}$, define the following $n-k+3$ hybrid experiments, where $\mathcal{H}_0(s)$ is the original $\mathsf{SS\text{-}ACD}(1^\lambda, |\psi\rangle, \mathsf{Adv}, s)$ experiment.

$\underline{\mathcal{H}_1(s)}$

In $\mathcal{H}_1(s)$, we sample the shares lazily using polynomial interpolation.

1. For each share i, sample the set of data indices $J_i \subset [t]$. Then for every share i and every check position $j \in [t] \backslash J_i$, sample the check position $|\psi_{i,j}\rangle$ as in $\mathcal{H}_0(s)$.

2. For each share i, divide the data indices J_i into a left set J_i^L of size ℓ and a right set J_i^R of size $t' - \ell$. For each $j \in J_i^L$, sample $f(it+j) \leftarrow \mathbb{K}$ uniformly at random.

3. Until $k-1$ shares are corrupted, i.e. $|C| = k-1$, run $\mathsf{Adv}(\{S_j\}_{j \in C}, \mathcal{R})$ as in $\mathsf{SS\text{-}ACD}$, with the following exception. Whenever Adv corrupts a new share by outputting $(c_a, \mathcal{R})$, finish preparing share c_a by sampling $f(c_a t + j) \leftarrow \mathbb{K}$ uniformly at random for every $j \in J_{c_i}^R$.

   At the end of this step, exactly $p = (k-1)t' + (n-k+1)\ell$ points of f have been determined, in addition to $f(0) = s$. This uniquely determines f.

[1] This projector defines the "deletion predicate" mentioned in the technical overview (Section 11.2).

4. Continue to run $\mathsf{Adv}(\{S_j\}_{j\in C}, \mathcal{R})$ as in SS-ACD, with the following exception. Whenever Adv corrupts a new share by outputting $(c_{k-1+b}, \mathcal{R})$, finish preparing c_{k-1+b} by interpolating the points in $J^R_{c_{k-1+b}}$ using share d_b and any other set of $p - t'$ points that have already been determined on f. Specifically, let

$$\mathsf{Int}_{k-1+b} \subset \{0\} \cup \bigcup_{m\in C} \{mt + j : j \in J_m\} \cup \bigcup_{m\notin C} \{mt + j : j \in J^L_m\}$$

be any set of $p + 1$ indices to be used in the interpolation, such that

$$\{d_b t + j : j \in J_{d_b}\} \subset \mathsf{Int}_{k-1+b}$$

For each $j \in J^R_{c_{k-1+b}}$, compute

$$f(c_{k-1+b}t + j) \leftarrow \mathsf{Interpolate}_p\left(c_{k-1+b}t + j, \{(m, f(m)) : m \in \mathsf{Int}_{k-1+b}\}\right)$$

See Section 12.3 for the definition of Interpolate.

Note that if SS-ACD does not abort in a round, $|C\backslash D| \leq k - 1$. In the round where Adv corrupts c_{k-1+i}, $|C| = k - 1 + i$, so d_i has already been determined.

$\underline{\mathcal{H}_2(s)}$

In $\mathcal{H}_2(s)$, we purify the share sampling using a register $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_n)$ which is held by the challenger. The challenger will maintain a copy of share i in register $\mathcal{C}_i = (\mathcal{C}_{i,1}, \ldots, \mathcal{C}_{i,t})$. Both $\mathcal{S}$ and $\mathcal{C}$ are initialized to $|0\rangle$ at the beginning of the experiment.

1. For each share i, sample the set of data indices $J_i \subset [t]$. Then for every share i and every check position $j \in [t]\backslash J_i$, prepare the state

$$\propto \sum_{y\in\mathbb{K}} |y\rangle^{\mathcal{S}_{i,j}} \otimes |y\rangle^{\mathcal{C}_{i,j}}$$

Measure $\mathcal{C}_{i,j}$ in the Hadamard basis to obtain $y_{i,j}$ for the verification key.

2. Divide each J_i into J^L_i and J^R_i as in $\mathcal{H}_1(s)$. For each $j \in J^L_i$, prepare the state

$$\propto \sum_{y\in\mathbb{K}} |y\rangle^{\mathcal{S}_{i,j}} \otimes |y\rangle^{\mathcal{C}_{i,j}}$$

3. Until $k - 1$ shares are corrupted, i.e. $|C| = k - 1$, run $\mathsf{Adv}(\{S_j\}_{j\in C}, \mathcal{R})$ as in SS-ACD, with the following exception. Whenever Adv corrupts a new share by outputting $(c_a, \mathcal{R})$, for every $j \in J^R_{c_a}$ prepare the state

$$\propto \sum_{y\in\mathbb{K}} |y\rangle^{\mathcal{S}_{c_a,j}} \otimes |y\rangle^{\mathcal{C}_{c_a,j}}$$

4. Continue to run $\mathsf{Adv}(\{S_j\}_{j\in C}, \mathcal{R})$ as in SS-ACD, with the following exception whenever Adv corrupts a new share by outputting $(c_{k-1+b}, \mathcal{R})$. Let Int_{k-1+b} be the set of indices to be used in interpolation for share c_{k-1+b}, as in $\mathcal{H}_1(s)$. For each $j \in J_{c_{k-1+b}}$, compute

$$\mathcal{C}_{c_{k-1+b},j} \leftarrow \mathsf{Interpolate}_p\left(c_{k-1+b}t + j, (mt + j, \mathcal{S}_{m,j})_{mt+j\in\mathsf{Int}_{k-1+b}}\right)$$

Finally, copy $\mathcal{C}_{c_{k-1+b},j}$ into $\mathcal{S}_{c_{k-1+b},j}$ in the computational basis, i.e. perform a controlled NOT operation with source register $\mathcal{C}_{c_{k-1+b},j}$ and target register $\mathcal{S}_{c_{k-1+b},j}$.

We emphasize that the timing of initializing each $\mathcal{S}_{i,j}$ is the same as in $\mathcal{H}_1(s)$. Note that since $\mathcal{H}_2(s)$ outputs either $\bot$ or Adv's view, register $\mathcal{C}$ never appears in the output of the experiment.

$\mathcal{H}_{2+i}(s)$ for $i \in [n-k+1]$

The only difference between $\mathcal{H}_{2+i}$ and $\mathcal{H}_{3+i}$ is that when the i'th share d_i is deleted in $\mathcal{H}_{3+i}$ (i.e. D reaches size i), the challenger performs a "deletion predicate" measurement on register $\mathcal{C}_{d_i}$. Specifically, let cert_{d_i} be the certificate output by Adv for share d_i. Immediately after verifying cert_{d_i} and adding d_i to D, the challenger measures the data positions in register $\mathcal{C}_{d_i}$ (i.e. register $(\mathcal{C}_{d_i,j})_{j\in J_{d_i}}$) with respect to the binary projective measurement $\{\Pi_{\mathsf{cert}_{k+i}}, I - \Pi_{\mathsf{cert}_{k+i}}\}$. If the measurement result is "reject" (i.e. $I - \Pi_{\mathsf{cert}_{k+i}}$), immediately output $\bot$ in the experiment. The difference between $\mathcal{H}_2$ and $\mathcal{H}_3$ is the same, for $i = 1$.

In addition to hybrids $\mathcal{H}_0$ through $\mathcal{H}_{3+n-k}$, we define a set of simulated experiments. Each Sim_i will be useful for reasoning about hybrid $\mathcal{H}_{2+i}$. Sim_i is similar to $\mathcal{H}_{2+i}$ except that all of the shares are randomized, whereas in $\mathcal{H}_{2+i}$, shares corrupted after c_{k-1+i} are interpolated.

Sim_i for $i \in [n-k+1]$

Run the $\mathsf{SS\text{-}ACD}(1^\lambda, |\psi\rangle, \mathsf{Adv}, s)$ experiment, with the following exceptions.

- Do *not* initialize $(\mathcal{S}_1, \ldots, \mathcal{S}_n, \mathsf{vk}) \leftarrow \mathsf{Split}_{\mathbb{S}}(1^\lambda, s)$ in step 1.
- Whenever Adv corrupts a new share by outputting $(c_a, \mathcal{R})$, prepare the state

$$\propto \sum_{\mathbf{y}\in\mathbb{K}^t} |\mathbf{y}\rangle^{\mathcal{S}_{c_a}} \otimes |\mathbf{y}\rangle^{\mathcal{C}_{c_a}}$$

  Then, sample the set of data indices $J_{c_a} \subset [t]$ of size t' and for each check index $j \in [t]\backslash J_{c_a}$ measure $\mathcal{C}_{c_a,j}$ in the Hadamard basis to obtain $y_{a,j}$ for the verification key.
- For the first i deletions d_b where $b \leq i$, immediately after the challenger verifies cert_{d_b} and adds d_b to D, it measures the data positions in register $\mathcal{C}_{d_b}$ with respect to the binary projective measurement $\{\Pi_{\mathsf{cert}_{d_b}}, I - \Pi_{\mathsf{cert}_{d_b}}\}$. If the measurement result is "reject", immediately output $\bot$ in the experiment.

Claim 16.2.2. *For every secret s,*

$$\mathsf{TD}[\mathcal{H}_0(s), \mathcal{H}_2(s)] = 0$$

Proof. It is sufficient to show that $\mathsf{TD}[\mathcal{H}_0(s), \mathcal{H}_1(s)] = 0$ and $\mathsf{TD}[\mathcal{H}_1(s), \mathcal{H}_2(s)] = 0$. The former is true by the correctness of polynomial interpolation (see Section 12.3). To see the latter, observe that steps 1, 2, and 3 in $\mathcal{H}_2(s)$ are equivalent to sampling a uniformly random state (in *any* basis) in register $\mathcal{S}_{i,j}$ by preparing a uniform superposition over the basis elements in $\mathcal{S}_{i,j}$, then performing a delayed measurement from $\mathcal{S}_{i,j}$ to $\mathcal{C}_{i,j}$ in that basis. Observe that steps 1, 2, and 3 in $\mathcal{H}_1(s)$ also sample uniformly random states in $\mathcal{S}_{i,j}$. Now consider step 4. In $\mathcal{H}_2(s)$, step 4 performs a (classical) polynomial interpolation using copies of points $(it+j, f(it+j))$ that are obtained by measuring $\mathcal{S}_{i,j}$. This is equivalent to directly interpolating using $\mathcal{S}_{i,j}$ if $\mathcal{S}_{i,j}$ contained a computational basis state, which is the case in $\mathcal{H}_1(s)$. □

We show that $\mathcal{H}_2$ has negligible trace distance from $\mathcal{H}_{3+n-k}$ in Claim 16.2.4. To prove Claim 16.2.4, we will need an additional fact which we show in Claim 16.2.3. Claim 16.2.3 will also show that the final hybrid $\mathcal{H}_{3+n-k}$ has zero trace distance from Sim_{n-k+1}, which is independent of the secret s.

Let $\mathcal{H}_i[c_a](s)$ denote the truncated game where $\mathcal{H}_i(s)$ is run until the end of the round where the a'th corruption occurs, i.e. when $|C|$ reaches a. At this point, $\mathcal{H}_i[c_a](s)$ outputs the adversary's register $\mathcal{R}$ and the set of corrupted registers $\{\mathcal{S}_j\}_{j\in C}$, unless the game has ended earlier (e.g. from an abort).[2] Let $\mathcal{H}_i[d_b](s)$ similarly represent the truncated game where $\mathcal{H}_i(s)$ is run until the end of the round where the b'th deletion occurs, i.e. when $|D|$ reaches b. Define $\mathsf{Sim}_i[c_a]$ and $\mathsf{Sim}_i[d_b]$ similarly.

Observe that after the n'th corruption in any hybrid experiment, the rest of the challenger's actions in the experiment is independent of the secret s. Therefore for every hybrid $\mathcal{H}_i$ and every pair of secrets (s_0, s_1),

$$\mathsf{TD}[\mathcal{H}_i[c_n](s_0), \mathcal{H}_i[c_n](s_1)] = \mathsf{TD}[\mathcal{H}_i(s_0), \mathcal{H}_i(s_1)]$$

Claim 16.2.3. *For every $i \in [0, n-k+1]$ and every secret s,*

$$\mathsf{TD}[\mathcal{H}_{2+i}[c_{k-1+i}](s), \mathsf{Sim}_i[c_{k-1+i}]] = 0$$

Combining this claim with the previous observation about the relation of a truncated experiment to its full version, it is clear that

$$\begin{aligned}\mathsf{TD}[\mathcal{H}_{3+n-k}(s_0), \mathcal{H}_{3+n-k}(s_1)] &= \mathsf{TD}[\mathcal{H}_{3+n-k}[c_n](s_0), \mathcal{H}_{3+n-k}[c_n](s_1)] \\ &= \mathsf{TD}[\mathsf{Sim}_{n-k+1}[c_n], \mathsf{Sim}_{n-k+1}[c_n]] \\ &= 0\end{aligned}$$

By Claim 16.2.4, we have

$$\mathsf{TD}[\mathcal{H}_2(s_0), \mathcal{H}_2(s_1)] \leq \mathsf{TD}[\mathcal{H}_{3+n-k}(s_0), \mathcal{H}_{3+n-k}(s_1)] + \mathsf{negl}(\lambda)$$

Therefore, combining claims Claim 16.2.2, Claim 16.2.3, and Claim 16.2.4, we have

$$\begin{aligned}\mathsf{TD}[\mathcal{H}_0(s_0), \mathcal{H}_0(s_1)] &\leq \mathsf{TD}[\mathcal{H}_{3+n-k}(s_0), \mathcal{H}_{3+n-k}(s_1)] + \mathsf{negl}(\lambda) \\ &\leq 0 + \mathsf{negl}(\lambda)\end{aligned}$$

which completes the proof. All that remains is to prove Claim 16.2.3, and Claim 16.2.4.

Proof of Claim 16.2.3. We proceed via induction. This is clearly true for $i = 0$, since the first $k-1$ shares to be corrupted are prepared as maximally mixed states in both $\mathcal{H}_2(s)$ and Sim.

Before addressing the case of $i > 0$, we define some notation for our specific application of interpolation. When preparing a share c_{k-1+i} after it is corrupted, the challenger interpolates evaluations of f into a register

$$\mathcal{C}^R_{c_{k-1+i}} := (\mathcal{C}_{c_{k-1+i},j})_{j\in J^R_{c_{k-1+i}}}$$

$\mathcal{C}^R_{c_{k-1+i}}$ consists of the right data positions of share c_{k-1+i} and contains $t' - \ell$ $\mathbb{K}$-qudits. To do the interpolation, the challenger uses evaluations of f contained in registers

$$\mathcal{C}'_{d_i} := (\mathcal{C}_{d_i,j})_{j\in J_{d_i}}$$

[2]The truncated version of the game outputs both the set of corrupted registers and $\mathcal{R}$, while the full version only outputs $\mathcal{R}$. In the full version, the adversary can move whatever information it wants into $\mathcal{R}$. However, the truncated game ends early, so the adversary may not have done this when the game ends. Outputting the corrupted registers directly ensures that they appear in the output in some form if the game does not abort.

and some other registers which we group as $\mathcal{I}$. $\mathcal{C}'_{d_i}$ consists of the data positions in share d_i and contains t' $\mathbb{K}$-qudits. Since polynomial interpolation is a linear operation over $\mathbb{K}$, the system immediately after c_{k-1+i} is prepared can be described as a state

$$\sum_{\substack{\mathbf{x_1}\in\mathbb{K}^{t'}\\ \mathbf{x_2}\in\mathbb{K}^{d-t'}}} \alpha_{\mathbf{x_1},\mathbf{x_2}} |\mathbf{x_1}\rangle^{\mathcal{C}'_{d_i}} \otimes |\mathbf{x_2}\rangle^{\mathcal{I}} \otimes |\mathbf{R_1 x_1} + \mathbf{R_2 x_2}\rangle^{\mathcal{C}^R_{c_{k-1+i}}} \otimes |\mathbf{R_1 x_1} + \mathbf{R_2 x_2}\rangle^{\mathcal{S}^R_{c_{k-1+i}}} \otimes |\phi_{\mathbf{x_1},\mathbf{x_2}}\rangle^{\mathcal{C}',\mathcal{S}',\mathcal{R}}$$

where $\mathbf{R_1} \in \mathbb{K}^{(t'-\ell)\times t'}$ and $\mathbf{R_2} \in \mathbb{K}^{(t'-\ell)\times(d+1-t')}$ are submatrices of the interpolation transformation, where $\mathcal{S}^R_{c_{k-1+i}}$ contains the copy of the evaluations in $\mathcal{C}^R_{c_{k-1+i}}$, where $\mathcal{C}'$ and $\mathcal{S}'$ respectively consist of the unmentioned registers of $\mathcal{C}$ and $\mathcal{S}$, and where $\mathcal{R}$ is the adversary's internal register.

Now we will show that the claim holds for $i+1$ if it holds for i. Define the following hybrid experiments.

- $\mathcal{H}_{3+i}[c_{k+i}]$: Recall that the only difference between $\mathcal{H}_{2+i}$ and $\mathcal{H}_{3+i}$ is an additional measurement made in the same round that the $(i+1)$'th share is deleted, i.e. when $|D|$ reaches $i+1$.
- $\mathcal{H}'_{3+i}[c_{k+i}]$: The only difference from $\mathcal{H}_{3+i}[c_{k+i}]$ occurs when the adversary requests to corrupt share c_{k+i}. When this occurs, the challenger prepares the right data positions of share c_{k+i} as the state
$$\propto \sum_{\mathbf{y}\in\mathbb{K}^{t'}} |\mathbf{y}\rangle^{\mathcal{C}^R_{c_{k+i}}} \otimes |\mathbf{y}\rangle^{\mathcal{S}^R_{c_{k+i}}}$$
- $\mathsf{Sim}_i[c_{k+i}]$: The only difference from $\mathcal{H}'_{3+i}[c_{k+i}]$ is that $\mathsf{Sim}_i[c_{k+i}]$ is run until c_{k+i} is corrupted, then the experiment is finished according to $\mathcal{H}'_{3+i}[c_{k+i}]$.

We first show that $\mathcal{H}'_{3+i}[c_{k+i}]$ and $\mathsf{Sim}_i[c_{k+i}]$ are close. By the inductive hypothesis,

$$\mathsf{TD}[\mathcal{H}_{2+i}[c_{k-1+i}](s), \mathsf{Sim}_{i-1}[c_{k-1+i}]] = 0$$

Note that $\mathcal{H}_{2+i}(s)$ and $\mathcal{H}'_{3+i}(s)$ are identical until the $(i+1)$'th deletion d_{i+1}. Similarly, Sim_{i-1} and Sim_i are identical until the $(i+1)$'th deletion. Therefore

$$\mathsf{TD}[\mathcal{H}'_{3+i}[d_i](s), \mathsf{Sim}_i[d_i]] = 0$$

Finally, observe that if the experiment does not abort, then d_i is deleted before c_{k+i} is corrupted (otherwise $|C\backslash D| = k$ during some round). Because of this, $\mathcal{H}'_{3+i}[c_{k+i}]$ and $\mathsf{Sim}[c_{k+i}]$ behave identically after the round where d_i is corrupted. Therefore

$$\mathsf{TD}[\mathcal{H}'_{3+i}[c_{k+i}], \mathsf{Sim}[c_{k+i}]] = 0$$

It remains to be shown that

$$\mathsf{TD}[\mathcal{H}_{3+i}[c_{k+i}], \mathcal{H}'_{3+i}[c_{k+i}]] = 0$$

The only difference between $\mathcal{H}_{3+i}[c_{k+i}]$ and $\mathcal{H}'_{3+i}[c_{k+i}]$ is at the end of the last round, where c_{k+i} is corrupted.[3] If the experiment reaches the end of this round without aborting, then d_i has already been corrupted, since otherwise at some point $|C\backslash D| = k$. Furthermore, the deletion predicate measurement

[3] In the definition of the SS-ACD experiment, the corruption set C is updated in between adversarial access to the corrupted registers, which occur once at the beginning of each round. The truncated game outputs according to the updated C, including $\mathcal{S}_{c_{k+i}}$.

on $\mathcal{C}_{d_i}$ must have accepted or else the experiment also would have aborted. It is sufficient to prove that the two experiments have 0 trace distance conditioned on not aborting.

In $\mathcal{H}'_{3+i}[c_{k+i}]$, if the experiment does not abort then its end state (after tracing out the challenger's $\mathcal{C}$ register) is

$$\sum_{\substack{\mathbf{x_1}\in\mathbb{K}^{t'}\\ \mathbf{x_2}\in\mathbb{K}^{d-t'}\\ x_3\in\mathbb{K}^{t'-\ell}}} |\alpha_{\mathbf{x_1},\mathbf{x_2}}|^2 \, |x_3\rangle\langle x_3|^{\mathcal{S}^R_{c_{k-1+i}}} \otimes \mathsf{Tr}^{\mathcal{C}'}\left[|\phi_{\mathbf{x_1},\mathbf{x_2}}\rangle\langle\phi_{\mathbf{x_1},\mathbf{x_2}}|^{\mathcal{C}',\mathcal{S}',\mathcal{R}}\right]$$

We now calculate the end state of $\mathcal{H}'_{3+i}[c_{k+i}]$, conditioned on it not aborting. In this case, the challenger for $\mathcal{H}'_{3+i}[c_{k+i}]$ prepares the next corrupted register $\mathcal{S}_{c_{k+i}}$ by a polynomial interpolation which uses registers $\mathcal{C}_{d_i}$ and $\mathcal{I}$. The deletion predicate measurement on $\mathcal{C}_{d_i}$ must have accepted to avoid an abort, so before performing the interpolation, the state of the system is of the form

$$|\gamma\rangle^{A,C',\mathsf{Int}_2,C_{d_{k+i}}} = \sum_{\mathbf{u}\in\mathbb{K}^{t'}:h_{\mathbb{K}}(u)<\ell/2} \alpha_{\mathbf{u}} \, |\psi_{\mathbf{u}}\rangle^{\mathcal{S},\mathcal{C}',\mathcal{I}} \otimes H^{\otimes t'\lceil\log_2(n+1)\rceil} \, |\mathbf{u}+\mathsf{cert}_{k+i}\rangle^{\mathcal{C}_{d_i}}$$

We will apply Theorem 13.0.1 with input size t' and output size $t'-\ell$ to show that share d_i contributes uniform randomness to the preparation of c_{k+i}. Observe that $\mathcal{C}_{d_i}$ contains t' $\mathbb{K}$-qudits and the interpolation target $\mathcal{C}^R_{c_{k+i}}$ contains $t'-\ell$ $\mathbb{K}$-qudits. Furthermore, $[R_1|R_2] \in \mathbb{K}^{(t'-\ell)\times(d+1)}$ is an interpolation matrix for a polynomial of degree p. By Fact 12.3.1, any $t'-\ell$ columns of $[R_1|R_2]$ are linearly independent. In particular, any $t'-\ell$ columns of R_1 are linearly independent. Finally, note that $(t'-(t'-\ell))/2 = \ell/2$. Therefore by Theorem 13.0.1, the state of the system after preparing register $\mathcal{C}^R_{c_{k+i}}$ and tracing out $\mathcal{C}_{d_i}$ when the experiment ends at the end of this round is

$$\sum_{\mathbf{x_3}\in\mathbb{K}^{t'-\ell}} \mathsf{Tr}^{\mathcal{C}_{d_i}}\left[|\gamma_{\mathbf{x_3}}\rangle\langle\gamma_{\mathbf{x_3}}|^{\mathcal{C},\mathcal{S},\mathcal{R}}\right]$$

where

$$|\gamma_{\mathbf{x_3}}\rangle = \sum_{\substack{\mathbf{x_1}\in\mathbb{K}^{t'}\\ \mathbf{x_2}\in\mathbb{K}^{d-t'}}} \alpha_{\mathbf{x_1},\mathbf{x_2}} \, |\mathbf{x_1}\rangle^{\mathcal{C}_{d_i}} \otimes |\mathbf{x_2}\rangle^{\mathcal{I}} \otimes |\mathbf{x_3}+\mathbf{R_2}\mathbf{x_2}\rangle^{\mathcal{C}_{c_{k+i}}} \otimes |\mathbf{x_3}+\mathbf{R_2}\mathbf{x_2}\rangle^{\mathcal{S}_{c_{k+i}}} \otimes |\varphi_{\mathbf{x_1},\mathbf{x_2}}\rangle^{\mathcal{C}',\mathcal{S}',\mathcal{R}}$$

After this round, $\mathcal{H}_{3+i}[c_{k+i}]$ ends and register $\mathcal{C}$ is traced out. This yields the state

$$\begin{aligned}
&\sum_{\mathbf{x_3}\in\mathbb{K}^{t'-\ell}} \mathsf{Tr}^{\mathcal{C}}\left[|\gamma_{\mathbf{x_3}}\rangle\langle\gamma_{\mathbf{x_3}}|^{\mathcal{C},\mathcal{S},\mathcal{R}}\right]\\
&= \sum_{\substack{\mathbf{x_1}\in\mathbb{K}^{t'}\\ \mathbf{x_2}\in\mathbb{K}^{d-t'}\\ \mathbf{x_3}\in\mathbb{K}^{t'-\ell}}} |\alpha_{\mathbf{x_1},\mathbf{x_2}}|^2 \, |\mathbf{x_3}+\mathbf{R_2}\mathbf{x_2}\rangle\langle\mathbf{x_3}+\mathbf{R_2}\mathbf{x_2}|^{\mathcal{S}^R_{c_{k+i}}} \otimes \mathsf{Tr}^{C'}\left[|\phi_{\mathbf{x_1},\mathbf{x_2}}\rangle\langle\phi_{\mathbf{x_1},\mathbf{x_2}}|^{\mathcal{C}',\mathcal{S}',\mathcal{R}}\right]\\
&= \sum_{\substack{\mathbf{x_1}\in\mathbb{K}^{t'}\\ \mathbf{x_2}\in\mathbb{K}^{d-t'}\\ \mathbf{x_4}\in\mathbb{K}^{t'-\ell}}} |\alpha_{\mathbf{x_1},\mathbf{x_2}}|^2 \, |\mathbf{x_4}\rangle\langle\mathbf{x_4}|^{\mathcal{S}^R_{c_{k+i}}} \otimes \mathsf{Tr}^{\mathcal{C}'}\left[|\phi_{\mathbf{x_1},\mathbf{x_2}}\rangle\langle\phi_{\mathbf{x_1},\mathbf{x_2}}|^{\mathcal{C}',\mathcal{S}',\mathcal{R}}\right]
\end{aligned}$$

where $\mathbf{x_4} = \mathbf{x_3} + \mathbf{R_2}\mathbf{x_2}$. This state is identical to the state at the end of $\mathcal{H}'_{3+i}[c_{k+i}]$ conditioned on the experiments not aborting.

□

Claim 16.2.4. *For every $i \in [0, n-k]$ and every secret s,*

$$\mathsf{TD}[\mathcal{H}_{2+i}(s), \mathcal{H}_{3+i}(s)] = \mathsf{negl}(\lambda)$$

Proof. The only difference between $\mathcal{H}_{2+i}(s)$ and $\mathcal{H}_{3+i}(s)$ is an additional deletion predicate measurement $\Pi_{\mathsf{cert}_{d_{i+1}}}$ during the round where d_{i+1} is corrupted. Say the deletion predicate accepts with probability $1-\epsilon$. Then the Gentle Measurement Lemma (Lemma 2.3.1) implies that, conditioned on the deletion predicate accepting, the distance between $\mathcal{H}_{2+i}(s)$ and $\mathcal{H}_{3+i}(s)$ is at most $2\sqrt{\epsilon}$. We upper bound the case where the deletion predicate rejects by 1 to obtain

$$\mathsf{TD}[\mathcal{H}_{2+i}(s), \mathcal{H}_{3+i}(s)] \leq (1-\epsilon)2\sqrt{\epsilon} + \epsilon$$

Thus, it is sufficient to show that $\epsilon = \mathsf{negl}(\lambda)$, i.e. the deletion predicate accepts with high probability on $\mathcal{C}_{d_{i+1}}$ in $\mathcal{H}_{3+i}$. To show this, we consider the following hybrids, and claim that the probability that the deletion predicate accepts on $\mathcal{C}_{d_{i+1}}$ is *identical* in each of the hybrids.

- $\mathcal{H}_{3+i}$
- $\mathcal{H}_{3+i}[d_{i+1}]$: The only difference is that the game ends after the round where d_{i+1} is deleted.
- $\mathsf{Sim}_{i+1}[d_{i+1}]$: Recall that the only difference between $\mathcal{H}_{3+i}$ and Sim_{i+1} is that every share j is prepared as the maximally entangled state

$$\sum_{\mathbf{x}\in\mathbb{K}^t} |\mathbf{x}\rangle^{C_j} \otimes |\mathbf{x}\rangle^{A_j}$$

- $\mathsf{Sim}'_{i+1}[d_{i+1}]$: The same as $\mathsf{Sim}_{i+1}[d_{i+1}]$, except that after preparing the maximally entangled state for each share j, we delay choosing J_j and measuring the check indices $\mathcal{C}_{i,j}$ for $j \in [t]\backslash J_j$. These are now done immediately after Adv deletes j by outputting $(\mathsf{cert}_j, j, \mathcal{R})$, and before the challenger verifies cert_j.

Observe that $\mathcal{H}_{3+i}$ and $\mathcal{H}_{3+i}[d_{i+1}]$ are identical until the deletion predicate measurement in the round where d_{i+1} is deleted, so the probability of acceptance is identical. By Claim 16.2.3,

$$\mathsf{TD}[\mathcal{H}_{3+i}[c_{k+i}](s), \mathsf{Sim}_{i+1}[c_{k+i}]] = 0$$

Share d_{i+1} is deleted before share c_{k+i} is corrupted in both $\mathcal{H}_{3+i}(s)$ and Sim_{i+1}, unless they abort. Therefore

$$\mathsf{TD}[\mathcal{H}_{3+i}[d_{i+1}](s), \mathsf{Sim}_{i+1}[d_{i+1}]] = 0$$

and the probability of acceptance is identical in $\mathcal{H}_{3+i}[d_{i+1}](s)$ and $\mathsf{Sim}_{i+1}[d_{i+1}]$. Finally, the probability of acceptance is identical in $\mathsf{Sim}'_{i+1}[d_{i+1}]$ because the register C is disjoint from the adversary's registers.

Thus, it suffices to show that $\epsilon = \mathsf{negl}(\lambda)$ in $\mathsf{Sim}'_{i+1}[d_{i+1}]$. Since $\mathbb{K}$ forms a vector space over $\mathbb{F}_2$, the certificate verification measurement and $\Pi_{\mathsf{cert}_{d_{i+1}}}$ are diagonal in the binary Fourier basis (i.e. the Hadamard basis) for every cert. Therefore the probability that Verify accepts $\mathsf{cert}_{d_{i+1}}$ but the deletion predicate measurement *rejects* $\mathcal{C}_{d_{i+1}}$ is

$$\epsilon = \Pr_{\substack{\mathsf{cert},\mathbf{y}\in\mathbb{K}^t \\ J\subset[t]:|J|=t'}} \left[\mathsf{cert}_{\overline{J}} = \mathbf{y}_{\overline{J}} \wedge \Delta_{\mathbb{K}}(\mathsf{cert}_J, \mathbf{y}_J) \geq \frac{\ell}{2}\right]$$

where J is the set of data indices for share d_{i+1}, where $\overline{J}$ is the set complement of J (i.e. the set of check indices for share d_{i+1}), and where $\Delta_{\mathbb{K}}(\mathsf{cert}_J, \mathbf{y}_J) = h_{\mathbb{K}}(\mathsf{cert}_J - \mathbf{y}_J)$ is the Hamming distance of cert_J from y_J. Here, the probability is taken over the adversary outputting a certificate cert for d_{k+i}, the challenger sampling a set of check indices $\overline{J}$, and the challenger measuring all of register $\mathcal{C}_{d_{i+1}}$ in the Hadamard basis to obtain $\mathbf{y} \in \mathbb{K}^t$.

This value can be upper bounded using Hoeffding's inequality, for any fixed cert and $\mathbf{y}$ with $\Delta_{\mathbb{K}}(\mathsf{cert}_J, \mathbf{y}_J) \geq \ell/2$. Note that the probability of acceptance is no greater than if the r check indices $\overline{J}$ are sampled *with* replacement. In this case, the expected number of check indices which do *not* match is

$$\geq \frac{\ell r}{2t} = \frac{t\log(\lambda)}{\lambda + (n-k+1)\log(\lambda)} \frac{(\lambda + (n-k+1)\log(\lambda))^2}{2t} \tag{16.11}$$

$$= \frac{\log(\lambda)}{2}(\lambda + (n-k+1)\log(\lambda)) \tag{16.12}$$

Therefore Hoeffding's inequality (Claim 12.2.1) implies that

$$\epsilon \leq 2\exp\left(\frac{-2\left(\frac{\log(\lambda)}{2}(\lambda + (n-k+1)\log(\lambda))\right)^2}{(\lambda + (n-k+1)\log(\lambda))^2}\right) \tag{16.13}$$

$$= 2\exp\left(-\frac{\log^2(\lambda)}{2}\right) \tag{16.14}$$

$$= \mathsf{negl}(\lambda) \tag{16.15}$$

□

Chapter 17

Tighter Parameters for the Threshold Construction

In this section, we give alternate parameter settings for the construction in Figure 16.2 that result in slightly smaller share sizes. The parameters are described in Figure 17.1. The main difference from Figure 16.1 is that r is slightly smaller, which also impacts t.

The construction in Figure 16.2 uses the following parameters.

- Each share consists of t total $\mathbb{K}$-registers, where

$$t = (k+1)r\left(1 + \frac{(n-k+1)\log(\lambda)}{\sqrt{r} - (n-k+1)\log(\lambda)}\right) + 1$$

- A share is divided into r check indices and $t' = t - r$ data indices, where

$$r = \lambda + (n-k+1)^2 \log^2(\lambda)$$

- ℓ intuitively represents an upper bound on the amount of information which is not destroyed when an adversary generates a valid deletion certificate for a share.

$$\ell = t\frac{\log(\lambda)}{\sqrt{r}}$$

  See the proof of Lemma 16.2.1 for a more precise usage of ℓ.

- The secret will be encoded in a polynomial of degree

$$d = (k-1)t' + (n-k+1)\ell$$

Figure 17.1: Alternate Parameters for Secret Sharing with Adaptive Certified Deletion

Lemma 17.0.1. *The construction in Figure 16.2 has reconstruction correctness with the parameters in Figure 17.1.*

Proof. The set $\{(it+j, y_{i,j})\}_{i\in P', j\in[t]}$ contains kt pairs which were obtained by measuring k shares. As mentioned in Section 12.3, if all but $e < (kt-d)/2$ of these pairs $(it+j, y_{i,j})$ satisfy $y_{i,j} = f(it+j)$, then $\mathsf{Correct}_{\mathbb{K},d}$ recovers the original polynomial f, where $f(0) = s$. The only points which do not satisfy this are the check positions, of which there are r per share, for a total of kr. Therefore for correctness, we require that

$$\begin{aligned} 2kr &< kt - d && (17.1)\\ &= kt - (k-1)(t-r) - (n-k+1)\ell && (17.2)\\ &= t + (k-1)r - (n-k+1)\ell && (17.3) \end{aligned}$$

Therefore $t - (n-k+1)\ell > (k+1)r$. Substituting $\ell = t\frac{\log(\lambda)}{\sqrt{r}}$ yields

$$\begin{aligned} t\left(1 - (n-k+1)\frac{\log(\lambda)}{\sqrt{r}}\right) &> (k+1)r && (17.4)\\ t &> (k+1)r\frac{1}{1-(n-k+1)\frac{\log(\lambda)}{\sqrt{r}}} && (17.5)\\ &= (k+1)r\frac{\sqrt{r}}{\sqrt{r}-(n-k+1)\log(\lambda)} && (17.6)\\ &= (k+1)r\left(1 + \frac{(n-k+1)\log(\lambda)}{\sqrt{r}-(n-k+1)\log(\lambda)}\right) && (17.7) \end{aligned}$$

Note that Equation (17.5) requires that $\left(1-(n-k+1)t\frac{\log(\lambda)}{\sqrt{r}}\right) > 0$. Since the number of check positions is $r = \lambda + (n-k+1)^2\log^2(\lambda)$, we have

$$1 - (n-k+1)\frac{\log(\lambda)}{\sqrt{\lambda+(n-k+1)^2\log^2(\lambda)}} > 1 - \frac{(n-k+1)\log(\lambda)}{(n-k+1)\log(\lambda)} = 0 \qquad (17.8)$$

Finally, observe that the choice of parameters in the construction satisfies these constraints. □

Lemma 17.0.2. *The construction in Figure 16.2 has adaptive certified deletion security with the parameters in Figure 17.1.*

Proof Sketch. The proof is almost the same as that of Lemma 16.2.1, except for the application of Hoeffding's inequality in Claim 16.2.4. The expected number of check indices which do not match becomes

$$\begin{aligned} \geq \frac{\ell r}{2t} &= \frac{t\log(\lambda)}{\sqrt{\lambda+(n-k+1)^2\log^2(\lambda)}}\frac{\lambda+(n-k+1)^2\log^2(\lambda)}{2t}\\ &= \frac{\log(\lambda)}{2}\sqrt{\lambda+(n-k+1)^2\log^2(\lambda)} \end{aligned}$$

Then Hoeffding's inequality implies

$$\begin{aligned}
\epsilon &\leq 2\exp\left(\frac{-2\left(\frac{\log(\lambda)}{2}\sqrt{\lambda+(n-k+1)^2\log^2(\lambda)}\right)^2}{\lambda+(n-k+1)^2\log^2(\lambda)}\right) \\
&= 2\exp\left(-\frac{\log^2(\lambda)}{2}\right) \\
&= \mathsf{negl}(\lambda)
\end{aligned}$$

□

Part III

Certified Deniability

In this part, we put forth an application-independent paradigm for certified deletion. We call this paradigm *certified deniability* in reference to its connections with the classical notion of deniability [DDN91, CDNO97, DNS98, Pas03]. Intuitively, certified deniability aims to embody the comprehensive philosophy

Once a user deletes the delegated information, it is as if they never received it in the first place.

In comparison with other notions of certified deletion, including those presented in this thesis, certified deniability offers a more wholistic approach to deletion. Previous definitions, such as obfuscation, are tailored carefully to the chosen primitive.

This tailoring can allow behaviors that not explicitly considered in the definition, but that should still be considered "attacks". For example, [MPY24] defines revocable signatures so that no adversary can simultaneously produce a valid certificate and a signatures that is valid *with respect to the honest verification procedure*. This does not rule out the possibility of an adversary simultaneously producing a valid certificate along with irrefutable proof of the message having been signed; in fact, we show that MPY24's construction admits such a strategy. Presumably, anyone who wants a signature to be deleted would desire that all evidence of their signing is removed.

In contrast, certified deniability uses a simulation-style approach that rules out even attacks that are not explicitly considered.

Chapter 18

Results

We study certified deniability for two primitives: signatures and non-interactive zero-knowledge arguments (NIZKs). We lay the definitional groundwork for adding certified deniability to these primitives and study what assumptions enable them.

Simulation-Based Deniability. To define certified deniability, we use the simulation paradigm to capture the comprehensive deletion philosophy mentioned before. We require that the state of the adversary after producing a valid deletion certificate can be simulated *without* having received a signature (or NIZK) in the first place. This gives us a guarantee of the form *"anything an adversary could learn from a deleted signature/NIZK, they could learn without it."*

To achieve this notion for signatures and NIZKs, we augment the Fiat-Shamir transform to add certified deniability in the quantum random oracle model (QROM).

Theorem 18.0.1 (Informal; theorem 22.0.4, theorem 22.0.2)**.** *There exist NIZKs with certified deniability in the QROM. Furthermore, if one-way functions exist, then there exist signatures with certified deniability in the QROM.*

Notably, this result bypasses [Pas03]'s impossibility for deniable NIZKs in the random oracle model. This can be seen as a unique application of quantum mechanics to force a user to "forget" information in a quantifiable way. For more discussion, see the technical overview.

Evidence Against Plain Model Constructions. Since our results are in the QROM, a natural next question is whether we can hope to achieve certified deniability (of either variety) in the *plain* model. Unfortunately, it appears that there are significant barriers to doing so. We show that if the security proof treats the adversary as a black-box, then it cannot hope to show either of the above notions.

Theorem 18.0.2 (Informal)**.** *There is no signature/NIZK with certified deniability in the plain model with a security proof that makes black-box use of the adversary.*

Thus, any valid proof of security for a plain-model construction must be non-black-box. Although there have been many improvements in non-black-box techniques in the past decade, e.g. [BKP18, BS20], the technique we use for the barrier is particularly amenable to obfuscation, which has also seen vast improvements recently [JLS21, JLS22, JLLW23]. We leave a more definitive answer to the question of certified deniability in the plain model to future work.

Chapter 19

Technical Overview

Evidence-Collection Strategies. We begin by discussing how an adversarial verifier can retain irrefutable evidence of a signature even after it has been revoked in the existing revocable signature constructions. We will consider MPY24's construction as an example, though the same approach works for other existing constructions (such as the public-key quantum money based approach). MPY24 begins by defining and constructing a new primitive called *2-tier tokenized signatures*. This primitive enables a simple construction for revocable signatures that can sign only the message "0". Using Lamport signatures, these can be extended to sign any message, but the key cannot be re-used to sign a second message ("no-query" security).

Finally, they achieve revocation security even when the adversary receives additional signatures ("multi-query" security) by having a single master signing key. When the signer wishes to sign a new message m, they generate a fresh one-time key pair $(\mathsf{sk}_R, \mathsf{vk}_R)$ together with a revocable signature $|\psi_R\rangle$. Then, the signer signs $\mathsf{vk}_R \| m$ under their global signing key to associate the new key with m.

The problem lies in this final step. Every signature on a message m consists of a *classical* signature σ on $\mathsf{vk}_R \| m$, together with the revocable signature $|\psi_R\rangle$. However, ignoring $|\psi_R\rangle$ completely, σ is already irrefutable proof that the signer signed m. (We note that this simply shows that their construction cannot satisfy our stronger notion and does not constitute an attack on the MPY construction as per their definition.)

19.1 Definitions: Simulation-Style

Since it is possible for constructions to satisfy prior definitions, but still admit evidence-collection attacks, we need new definitions. In this section, we provide an in-depth discussion of our definitions and their merits. We use a simulation-style definition, which offers a robustness even against classes of attacks that have not yet been explicitly considered. As demonstrated by the evidence-collection on MPY24's revocable signatures, such robustness is invaluable for avoiding subtleties that creative attackers might take advantage of.

Signatures. In certified deniability for signatures, we consider the following scenario that might happen in the real world:

1. The adversarial verifier V^* receives a public verification key vk.
2. V^* receives some signatures (possibly including on the same message multiple times) on some multi-set of message M over a period of time.

3. Over time, V^* outputs a list of certificates for some set of signatures. The signer decides whether to accept or reject each one.

To transform this scenario into a security experiment $\mathsf{Sig\text{-}CDen\text{-}Exp}_{V^*}(\mathsf{sk}, \mathsf{vk})$, we consider outputting V^*'s view along with some additional information. Let D be the set of messages associated with the signatures which the signer received valid certificates for. The experiment also outputs the multi-set of messages $M \backslash D$ which V^* for which received signatures but did not present a valid certificate.

Certified deniability requires that there exists a simulator Sim which receives vk and access to whichever signatures it wants, then produces a view that is indistinguishable from $\mathsf{Sig\text{-}CDen\text{-}Exp}_{V^*}(\mathsf{sk}, \mathsf{vk})$ *even when the list of messages* Sim *queries for is included in the output*:

$$\{(\mathsf{vk}, M \backslash D, \mathsf{Sig\text{-}CDen\text{-}Exp}(\mathsf{sk}, \mathsf{vk}))\}_{(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)} \approx \left\{(\mathsf{vk}, M_{\mathsf{Sim}}, \mathsf{Sim}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})\right\}_{(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)}$$

Since the real adversary gets "free signatures" on messages in D, but all of Sim's queries are recorded, Sim must be able to produce its view *without* receiving the additional signatures in D.

A useful property of this definition is that it hides the number of messages which have been signed and subsequently deleted. To illustrate this, consider a scenario where V^* asks for a single signature on m^*, then deletes it honestly; the corresponding simulator receives no signatures. Intuitively, to be indistinguishable, the simulator must somehow convince the adversary that it has seen a valid signature, without actually forging one. A more relaxed definition might make this task easier by giving the simulator a signature $\sigma_\perp$ on $\perp$. In this case, the simulator must only convince the adversary that $\sigma_\perp$ is in fact a signature on m^*. This corresponds to revealing that *some* message was signed, if not which one. However, because our definition does not give *any* signatures to the simulator, it requires that after deletion, a third party cannot even tell whether a single signature was given out.

NIZKs. Certified deniability for NIZKs is defined similarly. In the real execution $\mathsf{NIZK\text{-}CDen\text{-}Exp}$, the prover uses a witness w to prove the truth of some statement x. The simulator must reproduce the (adversarial) verifier's view given only x, without a witness w.[1] The reader may notice that this definition seems almost identical to the standard zero knowledge definition. Indeed, Pass [Pas03] points out that zero knowledge is inherently deniable in the plain model; it is indistinguishable whether the transcript given to you was the result of an honest execution, or the result of running the simulator. In the case where the statement is hard to decide, a simulator could even "prove" a false statement. Of course, NIZKs do not exist in the plain model [GO94]. Instead, we must consider the deniability of NIZKs in the random oracle model or the common reference string model.

Certified Deniability in the QROM. Traditionally for NIZKs in the random oracle model, the simulator is allowed to choose the random oracle, or similarly it is allowed to choose common reference string in the CRS model. However, Pass points out that in the real world, the random oracle is fixed once and for all. It is not realistic to believe that an arbitrary prover backdoored the random oracle. So, they could not have run the simulator.

Following Pass, we define certified deniability in the QROM so that the simulator does not have the ability to choose the random oracle. Specifically, in the ideal world, the simulator has access to a truly random oracle H, which also appears in the experiment output:

$$\{(H, \mathsf{NIZK\text{-}CDen}^H(x, w))\}_{H \leftarrow \mathcal{H}} \approx \{(H, \mathsf{Sim}^H(x))\}_{H \leftarrow \mathcal{H}}$$

[1] We remark that this only considers a single NIZK, in contrast with the signature definition which considered multiple signatures at a time. The difference between these is due to the need to sample a vk at the start of the signature experiment. Since NIZKs do not have secret information like this, they have better composeability properties.

where $\mathcal{H}$ is the set of all functions $\{0,1\}^{p(\lambda)} \to \{0,1\}^{q(\lambda)}$.

The inclusion of the original oracle H prevents a simulator from reprogramming the random oracle. For example, if it were to reprogram the oracle to $H'(y) = v'$, instead of $H(y) = v$, then a distinguisher who queries the real H on y would immediately catch it. Without programmability, the task of simulating a NIZK becomes much harder. In fact, Pass utilizes the non-programmability enforced by this definition to show that (in the classical setting) deniable NIZKs are in fact *impossible* even with a random oracle.

The Power of Forgetting. We show that it is possible to avoid this issue in the quantum setting. Quantum mechanics offers the intriguing capability to force the adversary to *forget* information it queried on when it produces a valid deletion certificate. This enables the simulator to internally pretend that $H(y) = v'$ without later being detected by the distinguisher querying the real H on y, since the distinguisher never learns y.

We remark that in real world, where H is heuristically implemented, the third party does not actually believe that the implementation changed to be $H(y) = v$. However, we can heuristically say that if they do not know y, then they do not know anything about $H(y)$, so they have no evidence against $H(y) = v$ in the actual implementation either.

Relation to Other Primitives with Certified Deletion. Our new simulation-based certified deletion approach is stronger than some existing revocability definitions, but equivalent to others. For example, in revocable encryption, the simulator could simply encrypt 0 instead of m, since the storage provider anyway does not have the decryption key before deletion time. However, the existing revocability definition for revocable signatures is weaker than our simulation definition.

Broadly, we can categorize revocable primitives as "passive" or "active". Passive primitives, such as encryption or commitments, do now allow a user who holds the revocable object to utilize it even before revocation. Because they do not reveal information before deletion, existing definitions are generally equivalent to their simulation-based counterparts. On the other hand, active primitives, such as signatures [MPY24], arguments [HMNY22], or obfuscated programs [BGK+24], allow the user who holds a revocable object to utilize it until revocation. For these primitives, extra care must be taken to control *exactly* what information is retained after revocation. The simulation-based approach ensures that *only* the information which the user is intended to receive (e.g. a single bit indicating whether the signature was valid) is retained after revocation.

Before-the-Fact Coercion. [CGV22] previously introduced the idea of "before-the-fact" coercion for deniable encryption. This notion considers a coercer who approaches a victim before the victim computes a ciphertext (e.g. to cast a vote), then forces them to encrypt a particular message by requiring them to produce evidence that they encrypted that message. Even with this modification, the coercer should not be able to identify whether the encryptor encrypted a desired message m, or something else. Coledangelo, Goldwasser, and Vazirani showed that this notion is classical impossible, but possible to achieve using quantum techniques.

Signatures and NIZKs with certified deniability are naturally deniable even against before-the-fact coercion of the verifier. Even if the victimized verifier is forced to use a special device or obfuscated program to collect evidence about the received signature/NIZK, this evidence is rendered useless after the verifier produces a valid certificate. Classically, this notion is not possible even in the random oracle model, since the auxiliary program could record any ROM input-output behavior when run by the victim, enabling the coercer to check its correctness later. However, in the quantum setting, any such record might prevent the verifier from outputting a valid certificate since the certificate generation requires "forgetting"

QROM queries. Furthermore, if the victim is forced to collect evidence anyway by not giving a valid certificate, then the authority can at least identify that they have been subject to coercion!

Does No-Query [MPY24] Satisfy the Simulation Definition? The reader might observe that the evidence-collection attack described earlier only applied to their multi-query-secure construction. For technical reasons related to a parallel repetition amplification step (see the next section), their no-query construction also allows evidence-collection.

19.2 Constructions

We now give an overview of our constructions, starting with signatures. For signatures, we will consider a selective notion of security where the adversary V^* must specify whether it later intends to delete each signature when it queries for that signature.

Starting Point. We start by recalling how [MPY24] constructs revocable signatures for the message "0" (i.e. 2-tier tokenized signatures). The signer samples a random pair of values (x_0, x_1) and a random phase $c \leftarrow \{0,1\}$. Then, they compute one-way function images $f(x_0)$ and $f(x_1)$ and output

$$\mathsf{vk} \coloneqq (f(x_0), f(x_1)), \quad \mathsf{sk} \coloneqq (x_0, x_1, c), \quad |\psi_{\mathsf{sig}}\rangle \coloneqq |x_0\rangle + (-1)^c |x_1\rangle$$

Given this, a verifier can measure $|\psi\rangle$ and check the result matches either $f(x_0)$ or $f(x_1)$. This operation can be done coherently to avoid disturbing the state, enabling the verifier to delete after it verifies. The deletion certificate is obtained by measuring $|\psi\rangle$ in the Hadamard basis, which results in a vector d such that $c = d \cdot (x_0 \oplus x_1)$. Using the secret key, the signer can verify this certificate.

[MPY24] shows that the adversary cannot produce both such a d and one of x_0 or x_1, except with probability $1/2$. Then, they show that repeating the scheme in parallel amplifies the difficulty of this task to a $\mathsf{negl}(\lambda)$ success rate; no adversary can obtain both a Hadamard basis measurement of every index and a string containing one element from every pair (x_0, x_1).

Attempt at a Fix. As discussed previously, a weakness in MPY24's multi-query construction is that it directly signs the message m under the global verification key. We can avoid this issue by directly associating the local verification key with m using $H(x\|m)$ instead of $f(x)$ in the key vk_m. This is binding on m, but if the adversary "forgets" x after deletion, then $H(x\|m)$ looks uniformly random and independent of m. Now we can safely sign vk_m using the master signing key to validate fresh signatures $|\psi_{\mathsf{vk}_m}\rangle$.

Of course, the underlying scheme only ensures $1/2$ "forgetfullness". We cannot amplify security using parallel repetition because even a single retained instance, which can be cheated with probability $1/2$, is still irrefutable evidence. Instead, we secret share m using the additive λ-of-λ threshold secret sharing scheme, obtaining $m_1, \ldots, m_\lambda$ such that $m_1 \oplus \cdots \oplus m_\lambda = m$, and sign each share m_i using the modified scheme:

$$\mathsf{Sign}\big(\mathsf{sk}, H\big(x_0^i \| m_i\big) \| H\big(x_1^i \| m_i\big)\big),\ m_i,\ |\psi_i\rangle \coloneqq |x_0^i\rangle + (-1)^{c_i} |x_1^i\rangle$$

If the adversary forgets even a single pair (x_i^0, x_i^1), they would be unable to verify m_i. However, changing any m_i changes the signed message m.

Proof Technique: Forgetful Local Programming. Although this construction turns out to not fully satisfy certified deniability, it will be helpful in demonstrating our new technique: forgetful local programming. The simulator will require a signature σ_0 on any message, say on "0" (note that giving this to the simulator leaks the number of messages signed, if not which ones). It will attempt to locally convince the verifier that this signature is actually for m.

To do so, it picks a random index i and computes $m'_i = m_i \oplus m$. If m'_i were swapped for m_i, then the secret-shared message becomes $(m_1 \oplus \cdots \oplus m_\lambda) \oplus m = 0 \oplus m = m$. To create this change, the simulator creates a modified oracle H', which swaps the behavior of $x^i_b \| m_i$ with $x^i_b \| m'_i$, i.e.

$$H'(x^i_b \| m'_i) \coloneqq H(x^i_b \| m_i)$$
$$H'(x^i_b \| m_i) \coloneqq H(x^i_b \| m'_i)$$

Then, it runs the adversarial verifier V^* using H' and σ_0. If H' were the true oracle, then σ_0 is actually a signature on m.

The crux of the argument comes down to showing that any third party distinguisher cannot distinguish whether it has access to H or H'. Since σ_0 is a signature on 0 under H, but is a signature on m under H', it therefore cannot tell which message σ_0 was originally associated with. To argue the crux, we will show that if V^* produces a valid certificate, then the distinguisher cannot find either x^i_0 or x^i_1.

The Need for a New Approach. Unfortunately, this approach only achieves $1/\mathsf{poly}(\lambda)$ security.[2] The issue is that the adversary could simply guess i, and is correct with inverse polynomial probability. If it deletes the other indices honestly, then it can keep one of x^i_0 and x^i_1 with probability $1/2$ even while providing a valid deletion certificate. Then, given x^i_b, the distinguisher could immediately detect the local reprogramming. Without the forgetful local programming technique, the tools of constructing certifiably deniable NIZKs becomes very close to what is available in the classical setting, where we know the task is impossible.

One might hope that by increasing the number of values in superposition, e.g. by constructing $\sum_i (-1)^{c_i} |x_i\rangle$ instead of $|x_0\rangle \pm |x_1\rangle$, the likelihood that the adversary can keep an x_i while simultaneously producing a valid deletion certificate can be decreased. While this is true, a superposition over many x_i requires a proportionate number of signed random oracle images $H(x_i \| m)$. Increasing the "forgetfulness" of the adversary to overwhelming probability would blow up the size of the signature super-polynomially.

Another issue is that this approach allows the adversary to collective evidence that *at least one* message was signed, if not which message it was. In other words, the approach does not ensure that the *number* of signatures is deleted. This appears in the security sketch as the requirement that the simulator receives a post-quantum signature σ_0, which it cannot forge on its own.

Idea 1: Subspace States. To increase the "forgetfulness" of the adversary beyond polynomial factors, we turn to subspace states [AC12a]. A subspace state $|A\rangle \propto \sum_{a \in A}$ is a uniform superposition over elements of an $\lambda/2$-dimensional subspace A of $\mathbb{F}_2^\lambda$. Using them, we can avoid the exponential blow-up by signing the random oracle images in superposition:

$$|\sigma_m\rangle \coloneqq \sum_{a \in A \setminus \{0\}} |a\rangle \otimes |\mathsf{Sign}(\mathsf{sk}, H(a \| m))\rangle$$

[2] Actually, this cannot be black-box reduced to the hardness of finding x^i_0 or x^i_1 discussed above, because of a large loss that occurs when extracting QROM queries. This problem can be fixed by outputting the whole signature as the certificate and doing some additional technical work, though we omit the details since it is subsumed by our later construction.

The verifier can check such a state by (1) coherently running the signature verification procedure in the computational basis and (2) coherently checking that the signed message matches $H(a\|m)$. To delete the signature, they can simply return the whole state. If the signing algorithm is deterministic, then the signer can check the certificate by uncomputing the signature and random oracle image, then checking that the certificate is now $|A\rangle$ using a projection onto $|A\rangle$. We note that it is possible to make the signing algorithm deterministic by coherently deriving its randomness from a PRF evaluated at a, e.g. $H(k\|a)$ for random k.

We argue that this deletion check enforces "forgetfulness". Due to the direct product hardness property [BS23], we know that given a random subspace state $|A\rangle$, it is hard to find both a vector in $A\backslash\{0\}$ and a vector in $A^\perp\backslash\{0\}$, even given access to an oracle which decides membership in A and $A^\perp$. The membership oracle can be used to check if the returned certificate indeed contains an intact copy of $|A\rangle$. If the signer is able to recover $|A\rangle$ from the certificate, then it could obtain a vector in $A^\perp$ by measuring it in the Hadamard basis. Whenever this happens, the verifier cannot also remember *any* vector in A, other than 0. Thus, we can use the forgetful local programming technique to obtain $\mathsf{negl}(\lambda)$ security loss.

Idea 2: Fiat-Shamir in Superposition. This still leaves the issue of leaking the number of signatures. To solve this, we use the Fiat-Shamir paradigm in superposition. Fiat-Shamir transforms a sigma protocol[3] into a signature scheme. The signer samples a random secret key sk and gives out $\mathsf{vk} = f(\mathsf{sk})$ as the verification key, where f is a one-way function. To sign a message me, the signer computes a sigma protocol proving knowledge of an sk matching vk. Fiat-Shamir uses the first message s_1 of the sigma protocol to derive the second message s_2 as $H(m\|\mathsf{vk}\|s_1)$.

Performing the Fiat-Shamir signature in superposition yields:

$$|\sigma_m\rangle \coloneqq \sum_{a\in A\backslash\{0\}} |a\rangle \otimes |s_1^a, s_2^a = H(a\|m\|\mathsf{vk}\|s_1), s_3^a\rangle$$

where (s_1^a, s_3^a) are the prover's messages in the sigma protocol using randomness $H(k\|a)$. Verification and deletion are defined similarly to the signature construction above.

Also similarly to the previous construction, if the certificate is valid, then the verifier must have "forgotten" every element of $A\backslash\{0\}$. Thus, even if it knew some transcript (s_1^a, s_2^a, s_3^a) of the sigma protocol, it could not prove to a third party that s_2^a was really derived using $H(a\|m\|\mathsf{vk}\|s_1)$.[4] Instead, the third party would suspect that s_2^a was chosen carefully to match a faulty s_1.

Crucially, by locally programming H at points that include some $a \in A$, the simulator can simulate every Fiat-Shamir transcript (s_1^a, s_2^a, s_3^a) *only using knowledge of* vk (or of the statement x). Thus, in the signature case, it no longer needs to receive anything signed under vk to do its job.

NIZKs. A useful consequence of using Fiat-Shamir to construct signatures is that the Fiat-Shamir transform can also be used for turning a sigma protocol into a NIZK. The construction is similar to the signature case, except that $s_2^a = H(a\|m\|\mathsf{vk}\|s_1^a)$ is replaced with $H(a\|x\|s_1^a)$, where x is the NP statement being proven.

[3] A sigma protocol is a 3-message public-coin argument of knowledge which is zero-knowledge against an honest verifier whose second message is known ahead of time.

[4] It is tempting to try to use a single first message s_1 and derive the second and third messages of the sigma protocol in superposition using a. However, this would lead to answering multiple challenges using the same first message, which may not be possible with a simulated s_1.

Other Technical Challenges. We briefly mention two additional technical challenges that appear in our construction. First, it is not immediately obvious that Fiat-Shamir can be simulated in superposition, even if it is post-quantum secure. Previous works have addressed such issues by using collapsing protocols [Unr16b] or small-range distributions [Zha12]. However, both of these techniques require collapsing the argument/signature to a large degree, which is at odds with deletion: if a superposition state is indistinguishable from a a measured state, we are almost back to the classical case and there is no way to delete! To avoid this issue, we use complexity leveraging to switch each of the superposed transcripts to be simulated, one at a time.

Second, it is not immediately obvious that Fiat-Shamir is sound in a structured superposition as above. The soundness of Fiat-Shamir in the quantum setting is a highly nontrivial task, but has been shown under certain conditions in the case where the resulting argument is classical [DFMS19, LZ19]. To argue soundness of Fiat-Shamir in superposition, we actually use coset states. We show that if the coset offset is not known, then the coset state appears to have been measured in the computational basis. Thus, we can treat our construction as having a classical argument/signature when proving soundness.

19.3 Black-Box Barriers to Plain Model Constructions.

Finally, we give an overview of the black-box barrier for the plain model. We model the simulator as having black-box access to the unitary dilation of the adversary and its inverse, but it may not directly access the adversary's internal registers, e.g. its auxiliary input.

Consider an adversary $\mathcal{A}$ and a distinguisher D which, as part of their auxiliary input, share a program that includes a key pair $(\widetilde{\mathsf{sk}}, \widetilde{\mathsf{vk}})$ for an internal (post-quantum) signature scheme. The program on some input an alleged signature $|\sigma\rangle$ for a message m under key vk, verifies it and if the check passes, it signs $m\|\mathsf{vk}$ using the internal signature key $\widetilde{\mathsf{sk}}$. This operation to create a proof that m was signed is gentle because of the correctness of the candidate scheme, so $\mathcal{A}$ can still generate a valid certificate using the honest deletion algorithm, after it has obtained a proof for the distinguisher that m was signed. Now it has produced both a valid certificate and a post-quantum signature on $m\|\mathsf{vk}$. The distinguisher can simply check the latter using $\widetilde{\mathsf{vk}}$.

Observe that in the real world, D will almost always obtain a valid signature on $m\|\mathsf{vk}$. On the other hand, the simulator cannot hope to extract such a signature using black-box access to $\mathcal{A}$, unless it is able to forge signatures for the candidate scheme.

On a technical level, the analysis requires generalizing a technique introduced by [BBBV97] for analyzing the behavior of an oracle algorithm with a reprogrammed *classical* oracle to handle oracles that do quantum computation instead. Roughly, we show that if an oracle algorithm is able to distinguish between oracle access to two unitaries U_0 and U_1, then outputting its query register at a random timestep produces a mixed state with noticeable probability mass on pure states $|\psi\rangle$ where $U_0|\psi\rangle$ and $U_1|\psi\rangle$ are proportionally far in trace distance. The generalized technique may be of independent interest.

Relation to Program Obfuscation. The above sketch considers a model where the simulator has black-box access to the unitary dilation of the adversary, but *cannot* access the adversary's auxiliary input, except indirectly by querying the adversary. We could also consider a model where the simulator *can* access the auxiliary input. In this case, if the auxiliary input consists of a ideally-obfuscated program which has the signing key hard-coded, then the simulator's access to the signing key becomes identical to the model where the auxiliary input cannot be directly accessed. The fact that ideal obfuscation causes the two models to become identical suggests that any plain model construction would need to use non-black-box techniques to bypass potentially obfuscated auxiliary input.

Avoiding the Barrier with QROM. The quantum random oracle bypasses the black-box issue for a few reasons. First, reprogramming a random oracle is, in a sense, a non-black-box operation. Second, the black-box issue arises because the simulator needs to somehow trick the adversary into thinking it has received a signature on m^*, without actually receiving a signature. However, in the plain model, the simulator does not seem to have more power than a real adversary, who should not be able to forge signatures. Introducing the ability to temporarily reprogram the oracle via the forgetful reprogramming technique restores the necessary asymmetry between the simulator and a real adversary.

19.4 Related Works

Concurrent Work. [AK24] concurrently proposed NIZKs with certified deletion. Their definition ensures that no man-in-the-middle can receive a NIZK and simultaneously produce a valid certificate along with a NIZK that is accepting with respect to the *honest verification procedure* unless they already know a witness. This definition has similarities to [MPY24]'s revocable signature definition in that it only considers third party verifiers who run the honest verification procedure. Their construction does not satisfy certified deniability, in part due to the use of a parallel amplification step (see the technical overview). They do achieve classical certificates in their construction, which we leave open for certified deniability.

Quantum Deniability. [CGV22] revisit the problem of deniable encryption in the quantum setting. In classical deniability, the encryptor should be able to produce "fake" proof to the adversarial coercer how a given ciphertext is actually an encryption of some other message. Coledangelo, Goldwasser, and Vazirani propose a uniquely quantum spin on the task: by computing the ciphertext, any explanation for it is destroyed. Although this has similarities to our setting, their result is quite different. They ensure that the third-party coercer never sees the explanation. In contrast, the adversarial verifier necessarily sees the "explanation" for signatures/NIZKs - the signature/NIZK itself - but later is forced to "forget" it.

Certified Deletion. Certified deletion was first proposed by Broadbent and Islam for encryption [BI20b]. It has since been generalized to a variety of other primitives, e.g. [HMNY22, HKM$^+$24, Por23, APV23, AKN$^+$23, BK23, BGK$^+$24, BR24, MPY24]. To the best of our knowledge, the only two works to have considered any notion of simulation in defining certified deletion are [HMNY22] and [BGK$^+$24]. [HMNY22] considers certified everlasting zero knowledge, which uses a simulation definition as a result of standard definitions for zero knowledge. [BGK$^+$24] consider a simulation-style definition of obfuscation with certified deletion in the structured oracle model as a side result, inspired by definitions of ideal obfuscation. In contrast, signatures are not typically considered to be a "simulation primitive", and NIZKs require either the random oracle or CRS models, which require special care for deniability.

Unclonability. Unclonability prevents an adversary from transforming an object (such as a program) into two functioning copies of the object [Aar09]. It is closely related to certified deletion, since a functional copy of the object can be considered as the "certificate". Previously, Goyal, Malavolta, and Raizes [GMR24] considered a related notion to certified deniability under the name of "strongly unclonable proofs". In a strongly unclonable proof, an adversarial man-in-the-middle (MiM) who receives a simulated proof of some (potentially false) statement x, then interacts with two sound verifiers to prove statements $\widetilde{x}_1$ and $\widetilde{x}_2$. Strong unclonability guarantees that no MiM can convince *both* verifiers of false statements. GMR showed that in general, strongly unclonable NIZKs do not exist. Fortunately, their techniques do not extend to certified deniability. GMR's impossibility relies on an interactive verification, during which the MiM can forward messages between the two verifiers. In certified deniability, the NIZK

is deleted before any messages reach the second verifier. In general, our definition is also more robust; for example, it rules out the possibility of the third-party verifier who accepts false statements with probability $1/2$ from being convinced by a deleted NIZK with even probability $1/2 + \epsilon$.

Chapter 20

Preliminaries

We write $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ to be the set of all functions $f : \mathcal{X} \to \mathcal{Y}$.

20.1 Quantum Computation

Quantum Oracles. We recall a result from [BBBV97] that aids in reasoning about reprogrammed oracles. Consider a quantum adversary who has quantum query access to one of two classical oracles H and H'. They bound the ability of the adversary to distinguish between the two oracles in terms of the amplitude with which it queries on (classical) inputs x where $H(x) \neq H'(x)$. As a simple corollary, if the adversary is able to distinguish the two oracles in a polynomial number of queries, then measuring one of its queries at random produces an x such that $H(x) \neq H'(x)$ with noticeable probability.

Lemma 20.1.1 ([BBBV97], Paraphrased)**.** *Let H and H' be oracles which differ on some set of inputs $\mathcal{X}$. Let $|\psi_i\rangle = \sum_y \alpha_{y,i} |\phi_{y,i}\rangle \otimes |y\rangle_Q$ be the state of $\mathcal{A}^H$ at time i, where Q is the query register. Let $|\psi'_i\rangle$ be the state of $A^{H'}$ at time i. Then for all $T \in \mathbb{N}$,*

$$\mathsf{TD}(|\psi_T\rangle, |\psi'_T\rangle) \leq \sqrt{T \sum_{i=1}^{T} \sum_{x^* \in \mathcal{X}} |\alpha_{x^*,i}|^2}$$

where TD *denotes the trace distance.*

Quantum Random Oracles. In the quantum random oracle model, all parties have access to an oracle $H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})$ implementing a random function. To ease notation, we implicitly pad inputs to the random oracle: given $x \in \mathcal{X}_1$, where $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, we denote $H(x\|\vec{0})$ as $H(x)$.

It will be useful to sometimes be able to derive large amounts of randomness from the oracle as if it were a PRF. The following lemma formalizes this treatment.

Lemma 20.1.2. *Let $H : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{Y}$ and $G : \mathcal{X}_2 \to \mathcal{Y}$ be random oracles. Define $H_k : \mathcal{X}_2 \to \mathcal{Y}$ by $H_k(v) \coloneqq H(k\|v)$. If $1/|\mathcal{X}_1| = \mathsf{negl}(\lambda)$, then*

$$\left\{ (O_H, O_{H_k}) : \begin{array}{c} H \leftarrow \mathsf{Func}(\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{Y}) \\ k \leftarrow \mathcal{X}_1 \end{array} \right\} \approx_c \left\{ (O_H, O_G) : \begin{array}{c} H \leftarrow \mathsf{Func}(\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{Y}) \\ G \leftarrow \mathsf{Func}(\mathcal{X}_2, \mathcal{Y}) \end{array} \right\}$$

where O_f denotes oracle access to a function f.

Proof. For any $k \in \mathcal{X}_1$, define $H_{k,G}$ as

$$H_{k,G}(x) = \begin{cases} G(x') & \text{if } x = k\|x' \text{ for some } x' \\ H(x) & \text{else} \end{cases}$$

The right distribution is identically distributed to

$$\left\{ (O_{H_{k,G}}, O_{H_k}) : \begin{array}{c} H \leftarrow \mathsf{Func}(\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{Y}) \\ G \leftarrow \mathsf{Func}(\mathcal{X}_2, \mathcal{Y}) \end{array} \right\}$$

By lemma 20.1.1, any distinguisher who distinguishes $(O_{H_{k,G}}, O_{H_k})$ from (O_H, O_{H_k}) with advantage ϵ in q queries can produce some x^* such that $O_{H_{k,G}}(x^*) \neq O_H(x^*)$ with probability ϵ^2/q^2. Whenever this occurs, $x^* = k\|x$ for some x. Since k is drawn uniformly at random from $\mathcal{X}_1$, it must be the case that $\epsilon^2/q^2 \leq 1/|\mathcal{X}_1| = \mathsf{negl}(\lambda)$. If q is $\mathsf{poly}(\lambda)$, then ϵ must be $\mathsf{negl}(\lambda)$. □

Z-Twirl. It is well-known that adding a random phase to a state is equivalent to measuring the state in the computational basis. Here we present a slightly generalized form of this.

Lemma 20.1.3. *Let $|\psi\rangle = \sum_{x\in\mathcal{X}} \alpha_x |x\rangle \otimes |\phi_x\rangle$ be a quantum state where $\mathcal{X}$ is a vector space. Denote $|\psi_s\rangle := \sum_{x\in\mathcal{X}} \alpha_x (-1)^{s\cdot x} |x\rangle \otimes |\phi_x\rangle$ for any $s \in \mathcal{X}$. Then*

$$\frac{1}{|\mathcal{X}|} \sum_{s\in\mathcal{X}} |\psi_s\rangle\langle\psi_s| = \sum_{x\in\mathcal{X}} |\alpha_x|^2 |x\rangle\langle x| \otimes |\phi_x\rangle\langle\phi_x|$$

Proof. We compute

$$\begin{aligned} \frac{1}{|\mathcal{X}|} \sum_{s\in\mathcal{X}^n} |\psi_s\rangle\langle\psi_s| &= \frac{1}{|\mathcal{X}|} \sum_{s\in\mathcal{X}} \alpha_{x_1}\overline{\alpha_{x_2}} \sum_{x_1,x_2\in\mathcal{X}} (-1)^{(x_1-x_2)\cdot s} |x_1\rangle\langle x_2| \otimes |\phi_{x_1}\rangle\langle\phi_{x_2}| \\ &= \frac{1}{|\mathcal{X}|} \sum_{x_1,x_2\in\mathcal{X}} \alpha_{x_1}\overline{\alpha_{x_2}} |x_1\rangle\langle x_2| \otimes |\phi_{x_1}\rangle\langle\phi_{x_2}| \sum_{s\in\mathcal{X}} (-1)^{(x_1-x_2)\cdot s} \\ &= \sum_{x\in\mathcal{X}} |\alpha_x|^2 |x\rangle\langle x| \otimes |\phi_x\rangle\langle\phi_x| \end{aligned}$$

□

20.2 Argument Systems

Definition 20.2.1 (Argument)**.** *An argument system (P, V) for an NP language $\mathcal{L}$ is a (potentially interactive) protocol between a prover P and a verifier V where P inputs a statement and a witness and V outputs accept or reject. It should satisfy two properties:*

- ***Correctness.*** *If w is a witness for $x \in \mathcal{L}$, then at the end of the execution, V outputs accept.*
- ***Soundness.*** *For every adversarial prover P^*, if $x \notin \mathcal{L}$, then at the end of an execution $\langle P^*, V\rangle$, V outputs reject with probability $1 - \mathsf{negl}(\lambda)$. If this holds for QPT P^*, we call the soundness computational. If it holds for unbounded P^*, we call the soundness statistical.*

If the argument system is non-interactive, we consider P and V to consist of single operations Prove *and* Verify.

Definition 20.2.2 (Zero-Knowledge)**.** *An argument system (P, V) for an NP language $\mathcal{L}$ is zero knowledge if there exists a QPT algorithm* Sim *such that for all QPT adversaries V^* with auxiliary input register R_{V^*} and all statement/witness pairs (x, w) where $x \in \mathcal{L}$,*

$$\{\langle P(x,w), V^*(\mathsf{R}_{V^*})\rangle\} \approx_c \{\mathsf{Sim}(V^*, \mathsf{R}_{V^*}\}$$

A NIZK is simply a zero-knowledge argument with only a single message (i.e. it is non-interactive).

Another useful property for argument systems to hold is to be a proof of knowledge. This ensures that the prover must "know" a witness for the statement it is proving. Since we do not directly use the proof of knowledge property in this work, we refer the reader to [Unr12, LZ19].

A frequently useful class of argument arises from sigma protocols. Sigma protocols are three round, public-coin arguments with relaxed zero-knowledge properties. In a public-coin protocol, the verifier's messages are truly random.

Definition 20.2.3 (Sigma Protocol)**.** *A Sigma protocol for an NP language $\mathcal{L}$ with relation $\mathsf{Rel}_{\mathcal{L}}$ is a three-message public-coin argument system $\Sigma = (P_1, P_3, \mathsf{Verify}_\Sigma)$ for $\mathcal{L}$ with the following properties:*

- ***Special Soundness.*** *There exists an extractor E which, given two transcripts (s_1, s_2, s_3) and (s_1, s_2', s_3') for the same statement x with the same first message s_1 and different verifier challenges $s_2 \neq s_2'$, extracts a witness for x.*
- ***Special Honest-Verifier Zero Knowledge (HVZK).*** *There exists a QPT simulator Sim_Σ such that for every $(x, w) \in \mathsf{Rel}_{\mathcal{L}}$ and every second message s_2,*

  $$\{(s_1, s_2, s_3) : s_1 \leftarrow P_1(x, w), s_3 \leftarrow P_3(x, w, s_2)\} \approx_c \{\mathsf{Sim}_\Sigma(x, s_2)\}$$

  We say Σ is ϵ-secure if no QPT distinguisher has greater than ϵ advantage in distinguishing these two distributions.

Fiat-Shamir Transform. The Fiat-Shamir transform [FS87, BG93] modifies a sigma protocol to become non-interactive by using a random oracle H. Specifically, the prover first computes $s_1 \leftarrow P_1(x, w)$, then computes the verifier's challenge $s_2 \leftarrow H(x \| s_1)$, and finally computes $s_3 \leftarrow P_3(x, w, s_2)$ and outputs (s_1, s_2, s_3). To verify a transcript, the verifier checks that $s_2 = H(s_1)$, then verifies the transcript using Verify_Σ. Given a sigma protocol Σ, we denote the Fiat-Shamir transform of Σ as $\mathsf{FS}^H_\Sigma(x, w)$. We extend the notation as $\mathsf{FS}^H_\Sigma(x, w; r)$ when specifying the prover's randomness r for Σ.

Although the analysis of Fiat-Shamir is more complicated in the quantum setting, a series of works have shown that, under mild assumptions on the sigma protocol, post-quantum Fiat-Shamir is sound even with quantum access to H, culminating in [DFMS19, LZ19].

Theorem 20.2.4 ([LZ19])**.** *If a post-quantum sigma protocol has (1) perfect completeness, (2) quantum proof of knowledge, and (3) unpredictable first messages, then the Fiat-Shamir heuristic gives a quantum NIZKPoK.*

Their result requires that the sigma protocol has unpredictable first messages, which requires that the probability of two executions having the same first message is negligible for all (x, w):

$$\Pr[s_1 = s_1' : s_1 \leftarrow P_1(x, w), s_2 \leftarrow P_1(x, w)] = \mathsf{negl}(\lambda)$$

Fiat-Shamir can also be used to create a signature scheme from a sigma protocol by using an NP instance x where finding the witness is hard (e.g. the image of a one-way function) as the public key and

its witness w as the secret key. A signature on a message m is obtained by computing $\mathsf{FS}_{\Sigma}^{H(m\|\cdot)}(x,w)$, where $H(m\|\cdot)$ denotes H with the first portion of the input fixed to m.

[DFMS19, LZ19] also showed that if the underlying sigma protocol is collapsing,[1] then the Fiat-Shamir transform gives a secure signature scheme in the quantum random oracle model.

Definition 20.2.5 (Collapsing for Sigma Protocols [Unr16a, DFMS19, LZ19, CMSZ22])**.** *We say a protocol $\langle P, V\rangle$ is* collapsing *if for every polynomial-size interactive quantum adversary P^* and polynomial-size quantum distinguisher $\mathcal{A}$,*

$$|\Pr[1 \leftarrow \mathsf{CollapseExp}(0, P^*, \mathcal{A})] - \Pr[1 \leftarrow \mathsf{CollapseExp}(1, P^*, \mathcal{A})]| \leq \mathsf{negl}(\lambda)$$

For $b \in \{0,1\}$, the experiment $\mathsf{CollapseExp}(b, P^*, \mathcal{A})$ is defined as follows:

1. *The challenger executes $\langle P^*, V\rangle$, storing the result in registers $(\mathsf{R}_1, \ldots, \mathsf{R}_n)$. It measures every register R_n in the computational basis.*
2. *The challenger coherently evaluates $V(|m_1, \ldots, m_n\rangle)$ and measures the result. If it is reject, the experiment aborts by outputting a random bit.*
3. *If $b = 0$, the challenger does nothing. If $b = 1$, the challenger measures R_n in the computational basis.*
4. *The challenger sends $(\mathsf{R}_1, \ldots, \mathsf{R}_n)$ to $\mathcal{A}$. The experiment outputs $\mathcal{A}$'s output bit.*

Theorem 20.2.6 ([LZ19])**.** *If a post-quantum sigma protocol is collapsing, then the Fiat-Shamir heuristic gives a secure post-quantum digital signature scheme in the quantum random oracle model.*

We note that natural sigma protocols satisfying the requirements of both theorems 20.2.4 and 20.2.6 are known. For instance, Unruh [Unr12] shows that a slight modification of Blum's Hamiltonian path argument is a quantum proof of knowledge. It also has perfect completeness and unpredictable first messages. Furthermore, if the commitments in the first round are implemented via a random oracle, then it has subexponential HVZK.

20.3 Revocable Signatures and NIZKs

Definition 20.3.1. *A digital signature is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with the following behavior:*

- $\underline{\mathsf{Gen}(1^\lambda)}$ *takes in the security parameter and outputs a key pair $(\mathsf{sk}, \mathsf{vk})$.*
- $\underline{\mathsf{Sign}(\mathsf{sk}, m)}$ *takes in a signing key sk and a message m, then outputs a (potentially quantum) signature σ.*
- $\underline{\mathsf{Verify}(\mathsf{vk}, \sigma, m)}$ *takes in a verification key vk, an alleged signature σ, and a message m, then outputs accept or reject.*

A digital signature must statisfy the following properties:

[1] [DFMS19] refers to this property as "computational unique responses".

- ***Correctness.*** *For every message m:*

$$\Pr\left[\mathsf{Accept} \leftarrow \mathsf{Verify}(\mathsf{vk}, \sigma, m) : \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

- ***Existential Unforgeability under Chosen Message Attack (EUF-CMA).*** *For all QPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} m \notin M \\ \wedge \\ \mathsf{Accept} \leftarrow \mathsf{Verify}(\mathsf{vk}, \mathsf{R}_{\mathsf{sig}}, m)\end{array} : \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

  where M is the list of messages that the adversary queries to the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

Revocable Signatures. A revocable signature scheme [MPY24] augments the signature scheme syntax with two additional algorithms Del and DelVer. Additionally, $\mathsf{Sign}(\mathsf{sk}, m)$ is modified to output both a signature register $\mathsf{R}_{\mathsf{sig}}$ and a deletion verification key dk. The new algorithms act as follows:

- $\underline{\mathsf{Del}(\mathsf{R}_{\mathsf{sig}})}$ takes in a register containing a signature, then outputs a certificate register $\mathsf{R}_{\mathsf{cert}}$.
- $\underline{\mathsf{DelVer}(\mathsf{dk}, \mathsf{R}_{\mathsf{cert}})}$ takes in the deletion verification key and a certificate register, then outputs accept or reject.

Additionally, a revocable signature scheme should satisfy deletion correctness and a notion of revocation security. We omit [MPY24]'s notion of revocation security here. Instead, we define certified deniability for (revocable) signatures in section 21.1.

Definition 20.3.2 (Deletion Correctness)**.** *For all messages m,*

$$\Pr\left[\mathsf{Accept} \leftarrow \mathsf{DelVer}(\mathsf{dk}, \mathsf{R}_{\mathsf{cert}}) : \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{R}_{\mathsf{sig}}, \mathsf{dk}) \leftarrow \mathsf{Sign}(\mathsf{sk}, m) \\ \mathsf{R}_{\mathsf{cert}} \leftarrow \mathsf{Del}(\mathsf{R}_{\mathsf{sig}})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Revocable NIZKs. Although these have not be explicitly defined before, they follow similar syntax to revocable signatures, so we include the description in this section. A revocable NIZK is augmented with two additional algorithms Del and DelVer, which act similarly to their signature counterparts. Additionally, a revocable NIZK must satisfy deletion correctness. We define certified deniability for (revocable) NIZKS in section 21.2.

Chapter 21

Definitions of Certified Deniability

21.1 Signatures

Deniable authentication was initially defined by Dwork, Naor, and Sahai [DNS98] using the simulation paradigm. Informally, a signature is deniable if it could be *simulated* by using only public information.

We follow a similar simulation-based paradigm which uses a real experiment $\mathsf{Sig\text{-}CDen}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(\mathsf{vk})$. This experiment is parameterized by a QPT adversary $\mathcal{A}$ who receives auxiliary input in register $\mathsf{R}_{\mathcal{A}}$ and a key pair $(\mathsf{sk}, \mathsf{vk})$. It is defined as follows.

1. $\mathcal{A}$ is initialized with register $\mathsf{R}_{\mathcal{A}}$ and vk. The challenger is initialized with sk.
2. $\mathcal{A}$ gets access to a signing oracle for sk. Each time the signing oracle is queried for some m_i (including duplicates), it adds m_i to a multi-set M and appends the deletion verification key dk_i for the signature it gives out to an internal log.
3. The adversary outputs a list of certificate registers $\{\mathsf{R}_{\mathsf{cert,i}}\}_i$ and register $\mathsf{R}_{\mathsf{out}}$.
4. For each i where $\mathsf{DelVer}(\mathsf{dk}_i, \mathsf{R}_{\mathsf{cert,i}}) = \mathsf{Accept}$, add m_i to a multi-set D.
5. Output $(M \backslash D,\ \mathsf{R}_{\mathsf{out}})$.

If working in an oracle model where the parties have access to an oracle H, then we denote the experiment as $\mathsf{Sig\text{-}CDen}^{H}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}$.

Definition 21.1.1 (Certified Deniability for Signatures: Plain Model)**.** *A revocable signature scheme (see section 20.3)* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Del}, \mathsf{DelVer})$ *is* certifiably deniable *if for every QPT adversary $\mathcal{A}$, there exists a QPT simulator* Sim *such that for every QPT adversary $\mathcal{A}$ with poly-size auxiliary input register* $\mathsf{R}_{\mathcal{A}}$*,*

$$\left\{(\mathsf{vk}, \mathsf{Sig\text{-}CDen}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(\mathsf{sk}, \mathsf{vk})) : (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)\right\}$$
$$\approx_c$$
$$\left\{(\mathsf{vk}, M_{\mathsf{Sim}}, \mathsf{Sim}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathcal{A}, \mathsf{R}_{\mathcal{A}}, \mathsf{vk})) : (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)\right\}$$

where M_{Sim} is the set of messages on which Sim *queries* $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

We say that the scheme has selective *certified deniability if this condition holds for the following modified* $\mathsf{Sig\text{-}CDen}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(\mathsf{sk}, \mathsf{vk})$ *experiment. At the start of the experiment, after receiving* vk*, the adversary*

declares the set of messages it will delete. Additionally, each time it queries the signing oracle for m_i, it declares whether it intends to delete the signature for m_i. At the end of the experiment, if the multi-set D does not match the set of declared signatures, then the experiment outputs $(M \backslash D, \bot)$ instead of $(M \backslash D, \mathsf{R}_{\mathcal{A}})$.

The set M acts as a way to restrict Sim from querying on signatures which $\mathcal{A}$ decides to delete (potentially after seeing vk and other signatures). $\mathcal{A}$ gets to remove signatures from M in the experiment output by deleting them, but all of Sim's queries are recorded.

In the selective case, the adversary *only* has to declare the messages/signatures it intends to delete. It is free to receive other signatures, even adaptively. However, if it does not delete the signatures which it says it will delete, then the experiment aborts.

Certified Deniability in the QROM. We may also define certified deniability in the CRS model or the QROM model. Following [Pas03]'s definition from the classical setting, a deniable simulator does *not* have the ability to program the global random oracle; this is enforced by sampling a fresh random oracle before the experiment and including its description in the output of the experiment. Thus, any simulator which attempts to pretend that the oracle had different behavior is will be caught.

Definition 21.1.2 (Certified Deniability for Signatures: QROM)**.** *A revocable signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Del}, \mathsf{DelVer})$ *is* certifiably deniable in the quantum random oracle model *if there exists a QPT simulator* Sim *such that for every QPT adversary $\mathcal{A}$ with poly-size auxiliary input register $\mathsf{R}_{\mathcal{A}}$,*

$$\left\{(O_H, \mathsf{vk}, \mathsf{Sig\text{-}CDen}^H_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(\mathsf{sk}, \mathsf{vk})) : \begin{array}{c} H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y}) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}^H(1^\lambda) \end{array}\right\}$$
$$\approx_c$$
$$\left\{(O_H, \mathsf{vk}, M_{\mathsf{Sim}}, \mathsf{Sim}^{H, \mathsf{Sign}^H(\mathsf{sk}, \cdot)}(\mathcal{A}, \mathsf{R}_{\mathcal{A}}, \mathsf{vk})) : \begin{array}{c} H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y}) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}^H(1^\lambda) \end{array}\right\}$$

where O_H denotes oracle access to H and M_{Sim} is the set of messages on which Sim queries $\mathsf{Sign}^H(\mathsf{sk}, \cdot)$.

We say that the scheme has selective *certified deniability in the QROM if this condition holds for the following modified $\mathsf{Sig\text{-}CDen}^H_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(\mathsf{sk}, \mathsf{vk})$ experiment. At the start of the experiment, after receiving vk, the adversary declares the set of messages it will delete. Additionally, each time it queries the signing oracle for m_i, it declares whether it intends to delete the signature for m_i. At the end of the experiment, if the multi-set D does not match the set of declared signatures, then the experiment outputs $(M \backslash D, \bot)$ instead of $(M \backslash D, \mathsf{R}_{\mathcal{A}})$.*

We show in chapter 22 how to obtain signatures with certified deniability by adding certified deniability to the Fiat-Shamir transformation.

A Composeable Definition. We also give a definition where the adversary only receives a single signature, but the definition has the ability to compose with itself. We give the definition in the QROM, but it can be straightforwardly moved to the plain model.

Define the following security game $\mathsf{Sig\text{-}CDen\text{-}1}^H_{\mathcal{A}(\mathsf{R}_{\mathcal{A}}}(\mathsf{sk}, \mathsf{vk}, m)$:

1. Sample $(|\sigma\rangle, \mathsf{dk}) \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and send $|\sigma\rangle$ to the adversary.
2. The adversary outputs a certificate register $\mathsf{R}_{\mathsf{cert}}$ and an output register $\mathsf{R}_{\mathsf{out}}$.
3. If $\mathsf{DelVer}(\mathsf{dk}, \mathsf{R}_{\mathsf{cert}}) = \mathsf{Accept}$, output $\mathsf{R}_{\mathsf{out}}$. Otherwise output $\bot$.

Definition 21.1.3 (Composeable Certified Deniability for Signatures)**.** *A revocable signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Del}, \mathsf{DelVer})$ *has* composeable certified deniability in the quantum random oracle model *if there exists a QPT simulator* Sim *such that for every QPT adversary* $\mathcal{A}$ *with poly-size auxiliary input register* $\mathsf{R}_{\mathcal{A}}$*,* every message m, and every $(\mathsf{sk}, \mathsf{vk})$ in the support of $\mathsf{Gen}(1^\lambda)$,

$$\left\{(O_H, \mathsf{vk}, \mathsf{Sig\text{-}CDen\text{-}1}^H_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(\mathsf{sk}, \mathsf{vk}, m)) : \ H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y}) \ \right\}$$
$$\approx_c$$
$$\left\{(O_H, \mathsf{vk}, \mathsf{Sim}^H(\mathcal{A}, \mathsf{R}_{\mathcal{A}}, \mathsf{vk}, m)) : \ H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y}) \ \right\}$$

where O_H *denotes oracle access to* H *and* M_{Sim} *is the set of messages on which* Sim *queries* $\mathsf{Sign}^H(\mathsf{sk}, \cdot)$.

There are a few main differences from Definition 21.1.2. First, the adversary only ever gets one signature, which they have promised to delete. Accordingly, the simulator gets *no* signatures/does not get access to a signing oracle. Second, the difference that makes this definition composeable is the quantification over all vk, rather than requiring that security holds over a randomly generated vk. As such, we can rely on deletion even when the adversary already holds information that depends on vk — such as another signature.[1] More formally, we show that the composeable definition implies the selective definition which explicitly gives multiple signatures to the adversary.

Proposition 21.1.4 (Composeability)**.** *Any revocable signature scheme with composeable certified deniability (Definition 21.1.3) also has selective certified deniability (Definition 21.1.2). This holds both in the QROM and in the plain model.*

Proof. Let Sim_+ be the simulator for the composeable definition. We show how to build a simulator Sim_s for the selective definition.

Any adversary $\mathcal{A}$ for the selective definition makes at most $q = \mathsf{poly}(\lambda)$ queries to the signing oracle. We define a series of q hybrid adversaries $\mathcal{A}_i$ which do not perform queries $\leq i$ which it intends to delete.

- Hyb_0 is the selective experiment run on the following adversary $\mathcal{A}_0$, except omitting the final check that D matches the pre-declared multi-set of messages. It still includes the check that $\mathcal{A}$ presents valid certificates for queries it declared that it would delete.

  $\mathcal{A}_0$ has sk hard-coded. It runs $\mathcal{A}(\mathsf{R}_{\mathcal{A}}, \mathsf{vk})$. During this, it answers all of $\mathcal{A}$'s queries using sk. Then, it checks the deletion certificates for the declared queries and outputs $\perp$ if any are rejected.

- Hyb_i is the selective experiment run on the following adversary $\mathcal{A}_i$, except omitting the final check that D matches the pre-declared multi-set of messages. It still includes the check that $\mathcal{A}$ presents valid certificates for queries it declared that it would delete.

  $\mathcal{A}_i^{\mathsf{Sign}(\mathsf{sk},\cdot)}$ has sk hard-coded and also has query access to $\mathsf{Sign}(\mathsf{sk}, \cdot)$. It runs $\mathcal{A}_{i-1}^{\mathsf{Sign}(\mathsf{sk},\cdot)}$ until $\mathcal{A}_{i-1}$ submits query i for m_i. Let $\mathcal{A}_{i-1}[i:]$ be the rest of its code and $\mathsf{R}_{\mathsf{i}-1}$ contain its current state. If $\mathcal{A}_{i-1}$ declares that it will delete the result of query i, run $\mathsf{Sim}_+^H(\mathcal{A}_{i-1}[i], \mathsf{R}_{\mathsf{i}-1}, m_i)$ by internally answering all of its queries to the signing oracle using sk. Otherwise, forward the i'th query to $\mathsf{Sign}(\mathsf{sk}, \cdot)$ and continue running $\mathcal{A}_{i-1}$ while answering all further queries using sk instead of directly querying $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

  Note that sk is *only* used for queries $\geq i + 1$.

[1] This quantification over all vk is possible because unforgeability is defined *separately* from certified deniability.

- $\mathsf{Sim}_s^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathcal{A})$ is the same as $\mathcal{A}_q$, except for the following changes. First, it does not use a hard-coded sk, since $\mathcal{A}_q$ does not require it either. Second, at the end of the game it checks that the pre-declared multi-set of messages matches the signatures for which it received a valid certificate; if not, it outputs $\bot$ instead of its view.

$\mathcal{A}_0$'s output is clearly identical to running the selective experiment with $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{R}_\mathcal{A}, \mathsf{vk})$ and omitting the final check that D matches the pre-declared multi-set of messages. Observe that if the composeable experiment on query i would abort for $\mathcal{A}_i$, then the selective experiment would also abort for $\mathcal{A}_i$. Therefore Hyb_i is indistinguishable from Hyb_{i-1} by the composeable security of the revocable signature scheme. Finally, $\mathsf{Sim}_s^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathcal{A})$ matches the full selective experiment by adding the check that D matches the pre-declared multi-set of messages.

The proof is essentially identical for the plain model. □

21.2 NIZKs

Previously, we saw a composeable definition for signatures which implies selective certified deletion when receiving multiple signatures. Since NIZKs are similar to signatures, we directly define certified deniability in the composeable style. If one were to define a similar multi-proof definition for NIZKs, then the following definition would imply the selective version of it in the same manner as for signatures.

Certified Deniability: Real Experiment. Certified deniability follows the standard real/ideal paradigm of simulator-based definitions. The real experiment $\mathsf{NIZK\text{-}CDen}_{\mathcal{A}(\mathsf{R}_\mathcal{A})}(s,w)$ is parameterized by an adversarial QPT algorithm with auxiliary input $\mathsf{R}_\mathcal{A}$ and some NP statement and witness pair (s,w). $\mathsf{NIZK\text{-}CDen}_{\mathcal{A}(\mathsf{R}_\mathcal{A})}(s,w)$ consists of the following distribution:

1. Sample $(|\pi\rangle, \mathsf{dk}) \leftarrow \mathsf{Prove}(s,w)$ and send $|\pi\rangle$ to the adversary.
2. The adversary outputs two registers $(\mathsf{R}_{\mathsf{cert}}, \mathsf{R}_\mathcal{A}) \leftarrow \mathcal{A}(\mathsf{R}_\mathcal{A}, |\pi\rangle)$.
3. If $\mathsf{DelVer}(\mathsf{dk}, \mathsf{R}_{\mathsf{cert}})$ outputs accept, then the experiment outputs the adversary's residual state register $\mathsf{R}_\mathcal{A}$. Otherwise, it outputs $\bot$.

If working in an oracle model where the parties have access to an oracle H, then we denote the experiment as $\mathsf{NIZK\text{-}CDen}^H_{\mathcal{A}(\mathsf{R}_\mathcal{A})}(s,w)$.

Definition 21.2.1 (Certified Deniability for NIZKs: Plain Model)**.** *A revocable non-interactive argument system (see section 20.3)* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{DelVer})$ *is* certifiably deniable *if there exists a QPT simulator* Sim *such that for every QPT adversary* $\mathcal{A}$ *with poly-size auxiliary input register* $\mathsf{R}_\mathcal{A}$ *and every NP statement/witness pair* (s,w)*,*

$$\left\{\mathsf{NIZK\text{-}CDen}_{\mathcal{A}(\mathsf{R}_\mathcal{A})}(s,w)\right\} \approx_c \left\{\mathsf{Sim}(s, \mathcal{A}, \mathsf{R}_\mathcal{A})\right\}$$

We may also define certified deniability in the CRS model or the QROM model. Following [Pas03]'s definition from the classical setting, a deniable simulator does *not* have the ability to program the global random oracle; this is enforced by sampling a fresh random oracle before the experiment and including its description in the output of the experiment. Thus, any simulator which attempts to pretend that the oracle had different behavior is likely to be caught.

Definition 21.2.2 (Certified Deniability for NIZKs: QROM)**.** *A revocable non-interactive argument system* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{DelVer})$ *is* certifiably deniable in the quantum random oracle model *if there exists a QPT simulator* Sim *such that for every QPT adversary* $\mathcal{A}$ *with poly-size auxiliary input register* $\mathsf{R}_{\mathcal{A}}$ *and every NP statement/witness pair* (s, w)*,*

$$\left\{\left(\mathsf{NIZK\text{-}CDen}^{H}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(s,w), O_H\right) : H \leftarrow \mathsf{Func}(\mathcal{X},\mathcal{Y})\right\}$$
$$\approx_c$$
$$\left\{\left(\mathsf{Sim}^{H}(s,\mathsf{Adv},\mathsf{R}_{\mathcal{A}}), O_H\right) : H \leftarrow \mathsf{Func}(\mathcal{X},\mathcal{Y})\right\}$$

where O_H *denotes oracle access to* H*.*

We show in chapter 22 how to obtain NIZKs with certified deniability by adding certified deniability to the Fiat-Shamir transformation.

Chapter 22

Fiat-Shamir with Certified Deniability

In this section, we show how to modify the Fiat-Shamir transform to add certified deniability. As a result of this general paradigm, we obtain both signatures with certified deniability (see section 21.1) and non-interactive zero knowledge with certified deniability (see section 21.2).

Let Σ be a Sigma protocol, and denote FS_Σ^H as its Fiat-Shamir transform with oracle access to H. We denote an oracle with the first part of the input fixed to v as $H(v\|\cdot)$. On query w, it returns $H(v\|w)$.

Construction 22.0.1 (NIZK with Certified Deniability)**.** *The construction, which we call* $\mathsf{FS\text{-}CDen}_\Sigma$ *for easy reference later, is as follows.*

- $\underline{\mathsf{Prove}(x,w)}$*:*
  1. *Sample a subspace* $A \subset \{0,1\}^\lambda$ *of dimension* $\lambda/2$ *and an offset* $s \leftarrow \{0,1\}^\lambda$*, then prepare the coset state* $|A_{0,s}\rangle \propto \sum_{a\in A}(-1)^{a\cdot s}|a\rangle$ *in register* $\mathsf{R_A}$*. Initialize register* $\mathsf{R_\Sigma} = (\mathsf{R}_{\Sigma,1},\mathsf{R}_{\Sigma,2},\mathsf{R}_{\Sigma,3})$ *to* $|0\rangle$*.*
  2. *Sample a key* $k \leftarrow \{0,1\}^\lambda$ *and apply the isometry*
  $$|a\rangle_{\mathsf{R_A}} \otimes |\vec{0}\rangle_{\mathsf{R_\Sigma}} \mapsto |a\rangle_{\mathsf{R_A}} \otimes |\mathsf{FS}_\Sigma^{H(a\|\cdot)}(x,w;H(k\|a))\rangle_{\mathsf{R_\Sigma}}$$
  to registers $\mathsf{R_A}$ *and* $\mathsf{R_\Sigma}$*. This results in a state*
  $$|\pi\rangle :\propto \sum_{a\in A}(-1)^{s\cdot a}|a\rangle_{\mathsf{R_A}} \otimes |s_1^a, s_2^a, s_3^a\rangle_{\mathsf{R_\Sigma}}$$
  where $s_2^a = H(a\|s_1^a)$ *and* (s_1^a, s_3^a) *are the prover's messages from* Σ *using randomness* $H(k\|a)$*.*
  3. *Output* $|\pi\rangle$ *as the argument and* $\mathsf{dk} := (A,k,w,s)$ *as the deletion key.*
- $\underline{\mathsf{Verify}(x,\mathsf{R_{arg}})}$*:*
  1. *Parse* $\mathsf{R_{arg}} = (\mathsf{R_A},\mathsf{R_\Sigma})$*.*
  2. *Coherently evaluate* $\mathsf{FS}_\Sigma.\mathsf{Verify}$ *on register* $\mathsf{R_\Sigma}$*, then measure and output the result.*
- $\underline{\mathsf{Del}(\mathsf{R_{arg}})}$*: Output* $\mathsf{R_{arg}}$*.*
- $\underline{\mathsf{DelVer}(\mathsf{dk},\mathsf{R_{cert}})}$*:*
  1. *Parse* $\mathsf{dk} = (A,k,w)$*. Parse* $\mathsf{R_{arg}} = (\mathsf{R_A},\mathsf{R_\Sigma})$*.*

2. *Apply the isometry*

$$|a\rangle_{\mathsf{R_A}} \otimes |y\rangle_{\mathsf{R_\Sigma}} \mapsto |a\rangle_{\mathsf{R_A}} \otimes |y \oplus \mathsf{FS}_{\Sigma}^{H(a\|\cdot)}(x, w; H(k\|a))\rangle_{\mathsf{R_\Sigma}}$$

3. *Measure register* $\mathsf{R_A}$ *with respect to the PVM* $\{|A_{0,s}\rangle\langle A_{0,s}|, I - |A_{0,s}\rangle\langle A_{0,s}|\}$. *Output accept if the result is the former, and reject if it is the latter.*

Theorem 22.0.2. *If* Σ *is a* $2^{-\lambda}$*-secure sigma protocol for a language* $\mathcal{L}$ *with (1) perfect completeness and (2) quantum proof of knowledge, then* $\mathsf{FS\text{-}CDen}_{\Sigma}$ *(Construction 22.0.1) is a NIZKPoK for* $\mathcal{L}$ *with certified deniability in the QROM.*

Remark: As mentioned in section 20.2, such sigma protocols exist unconditionally in the QROM.

Proof. We prove certified deniability in claim 22.1.1. Correctness of deletion follows from inspection and the fact that $|A_{0,s}\backslash\{0\}\rangle$ is negligibly close in trace distance to $|A_{0,s}\rangle$.

By lemma 20.1.3, any NIZK $|\pi\rangle$ is perfectly identical, from the verifier's point of view, from whether the first register $\mathsf{R_A}$ has been measured in the computational basis. Thus, construction 22.0.1 is sound and zero-knowledge if sampling a random a and outputting $a\|\mathsf{FS}^{H(a\|\cdot)}(x, w)$ is sound and zero-knowledge. Observe that a can be considered to be part of the first message of underlying sigma protocol, which makes the sigma protocol have unpredictable first messages. Then theorem 20.2.4 implies that $a\|\mathsf{FS}^{H(a\|\cdot)}(x, w)$ is a NIZKPoK, which implies it is sound and zero-knowledge. □

Construction 22.0.1 is already a NIZK, if instantiated with an appropriate sigma protocol. To obtain a signature scheme, a few more details are needed.

Construction 22.0.3 (Signature with Certified Deniability)**.** *Let* $f : \{0,1\}^{\lambda} \mapsto \{0,1\}^{\mathsf{poly}(\lambda)}$ *be a one-way function and let* Σ *be a sigma protocol for the language*

$$\mathcal{L}_f = \left\{x \in \{0,1\}^{\mathsf{poly}(\lambda)} : \exists w \in \{0,1\}^{\lambda} \text{ s.t. } f(w) = x\right\}$$

- $\underline{\mathsf{Gen}(1^{\lambda})}$: *Sample* $w \leftarrow \{0,1\}^{\lambda}$. *Output* $\mathsf{vk} = f(w)$ *and* $\mathsf{sk} = w$.
- $\underline{\mathsf{Sign}(\mathsf{sk}, m)}$: *Evaluate* $(|\pi\rangle, \mathsf{dk}) \leftarrow \mathsf{FS\text{-}CDen}_{\Sigma}.\mathsf{Prove}^{H(m\|\cdot)}(\mathsf{vk}, \mathsf{sk})$ *and output the result.*
- $\underline{\mathsf{Verify}(\mathsf{vk}, m, \mathsf{R_{arg}})}$: *Evaluate* $\mathsf{FS\text{-}CDen}_{\Sigma}.\mathsf{Verify}^{H(m\|\cdot)}(\mathsf{vk}, \mathsf{R_{arg}})$.
- $\underline{\mathsf{Del}(\mathsf{R_{arg}})}$: *Output* $\mathsf{FS\text{-}CDen}_{\Sigma}.\mathsf{Del}^{H(m\|\cdot)}(\mathsf{R_{arg}})$.
- $\underline{\mathsf{DelVer}(\mathsf{dk}, \mathsf{R_{cert}})}$: *Output* $\mathsf{FS\text{-}CDen}_{\Sigma}.\mathsf{DelVer}^{H(m\|\cdot)}(\mathsf{dk}, \mathsf{R_{cert}})$.

Theorem 22.0.4. *If* f *is a one-way function and* Σ *is a collapsing,* $2^{-\lambda}$*-secure sigma protocol for* $\mathcal{L}_f$*, then construction 22.0.3 is a signature scheme with selective certified deniability in the QROM.*

Proof. We prove certified deniability in claim 22.1.1. Correctness of deletion follows from inspection and the fact that $|A_{0,s}\backslash\{0\}\rangle$ is negligibly close in trace distance to $|A_{0,s}\rangle$.

By lemma 20.1.3, any NIZK $|\pi\rangle$ is perfectly identical, from the verifier's point of view, from whether the first register $\mathsf{R_A}$ has been measured in the computational basis. Thus, construction 22.0.1 is existentially unforgeable if sampling a random a and outputting $a\|\mathsf{FS}_{\Sigma}^{H(m\|a\|\cdot)}(x, w)$ is existentially unforgeable. Observe that a can be considered to be part of the first message of the underlying sigma protocol without affecting its properties. Then theorem 20.2.6 implies that $a\|\mathsf{FS}_{\Sigma}^{H(m\|a\|\cdot)}(x, w)$ is existentially unforgeable. □

22.1 Proof of Certified Deniability

Claim 22.1.1. *If* Sigma *is a* $2^{-\lambda}$*-secure sigma-protocol for a language* $\mathcal{L}$*, then Construction 22.0.1 satisfies certified deniability for NIZKs in the QROM (definition 21.2.2). Furthermore, assuming the same, Construction 22.0.3 satisfies selective certified deletion for signatures in the QROM (definition 21.1.2).*

Proof. The proofs of the two sub-claims are almost identical, except for a slight difference in the simulators. The NIZK simulator Sim uses statement x and is given below. The signature simulator uses the verification key vk instead of x, and additionally forwards any adversarial queries for signatures to the signing oracle, except if the adversary would query for m^*. Furthermore, in the case of signatures, we regard $H(m\|\cdot)$ to be the random oracle, so we omit explicitly prepending m to the random oracle queries below.

On input a statement x, the adversary's code $\mathcal{A}$ and auxiliary input register $\mathsf{R}_{\mathcal{A}}$, the simulator Sim works as follows:

1. Sample a subspace $A \subset \{0,1\}^\lambda$ of dimension $\lambda/2$ and sample an offset $s \leftarrow \{0,1\}^\lambda$,then prepare the coset state $|A_{0,s}\backslash\{0\}\rangle \propto \sum_{a\in A\backslash\{0\}}(-1)^{a\cdot s}|a\rangle$ in register $\mathsf{R_A}$. Initialize register $\mathsf{R}_\Sigma = (\mathsf{R}_{\Sigma,1},\mathsf{R}_{\Sigma,2},\mathsf{R}_{\Sigma,3})$ to $|0\rangle$.

2. Sample keys $k_{\mathsf{ch}}, k \leftarrow \{0,1\}^\lambda$. Apply the isometry

$$|a\rangle_{\mathsf{R_A}} \otimes |0\rangle_{\mathsf{R}_\Sigma} \mapsto |a\rangle_{\mathsf{R_A}} \otimes |\mathsf{Sim}_\Sigma(x, H(k_{\mathsf{ch}}\|a); H(k\|a)\rangle_{\mathsf{R}_\Sigma}$$

   to registers $\mathsf{R_A}$ and R_Σ.

$$|\widetilde{\pi}\rangle :\propto \sum_{a\in A\backslash\{0\}} (-1)^{a\cdot s}|a\rangle_{\mathsf{R_A}} \otimes |\widetilde{s_1^a}, \widetilde{s_2^a}, \widetilde{s_3^a}\rangle_{\mathsf{R}_\Sigma}$$

   where $\widetilde{s_2^a} = H(k_{\mathsf{ch}}\|a)$ and $(\widetilde{s_1^a}, \widetilde{s_3^a})$ are obtained by running the honest verifier zero knowledge simulator Sim_Σ for Σ.

3. Define the random oracle H' as follows:

$$H'(q_1\|q_2\|q_3) \coloneqq \begin{cases} H(k_{\mathsf{ch}}\|q_1) & \text{if } q_1 \in A\setminus\{0\} \text{ and } q_2\|q_3 = x\|\widetilde{s_1^{q_1}} \\ H(q_1\|q_2\|q_3) & \text{else.} \end{cases} \tag{22.1}$$

   where $\left(\widetilde{s_1^a}, \widetilde{s_2^a}, \widetilde{s_3^a}\right) = \mathsf{Sim}_\Sigma(x, H(k_{\mathsf{ch}}\|a); H(k\|a))$ for all $a \in A\backslash\{0\}$. Note that this may be efficiently evaluated on any computational basis queries using the description of A, and thus on any quantum queries in general.

4. Compute the adversary's output $(\mathsf{R}_{\mathsf{cert}}, \mathsf{R}_{\mathcal{A}}) \leftarrow \mathcal{A}^{H'}(\mathsf{R}_{\mathcal{A}}, |\widetilde{\pi}\rangle)$.

5. Parse $\mathsf{R}_{\mathsf{cert}} = (\mathsf{R_A}, \mathsf{R}_\Sigma)$ and compute the isometry

$$|a\rangle_{\mathsf{R_A}} \otimes |y\rangle_{\mathsf{R}_\Sigma} \mapsto |a\rangle_{\mathsf{R_A}} \otimes |y \oplus \mathsf{Sim}_\Sigma(x, H(k_{\mathsf{ch}}\|q_2); H(k\|a))\rangle_{\mathsf{R}_\Sigma}$$

   on registers $\mathsf{R_A}$ and R_Σ.

6. Measure register $\mathsf{R_A}$ with respect to the PVM $\{|A_{0,s}\rangle\langle A_{0,s}|, I - |A_{0,s}\rangle\langle A_{0,s}|\}$. If the result is the former, output $\mathsf{R}_{\mathcal{A}}$. Otherwise output $\perp$.

We now show that this simulator satisfies definition 21.2.2, i.e. the joint distribution over the output of the simulator and the description of H is indistinguishable from the real certified deniability experiment. Consider the following hybrid experiments:

- $\underline{\mathsf{Hyb}_0(w)}$: The real certified deniability experiment $\mathsf{NIZK\text{-}CDen}^{H}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(x,w)$. Note that this does not encompass the whole output distribution from the certified deniability definition in the QROM (definition 21.1.2); it is missing the O_H output.

- $\underline{\mathsf{Hyb}_1(w)}$: The only difference from Hyb_1 is that we replace the oracle H with a reprogrammed oracle H'_1, defined as[1]

$$H'_1(q_1\|q_2\|q_3) := \begin{cases} H(k_{\mathsf{ch}}\|q_1) & \text{if } q_1 \in A\backslash\{0\} \text{ and } q_2\|q_3 = x\|s_1^{q_1} \\ H(q_1\|q_2\|q_3) & \text{else.} \end{cases} \tag{22.2}$$

  where $s_1^a = P_{\Sigma,1}((x,w); H(k\|a))$.[2] In the security game $\mathsf{Hyb}_1(w) = \mathsf{NIZK\text{-}CDen}^{H'_1}_{\mathcal{A}(\mathsf{R}_{\mathcal{A}})}(x,w)$, this results in a NIZK

$$|\pi\rangle \propto \sum_{a\in A\backslash\{0\}} (-1)^{a\cdot s} |a\rangle \otimes |s_1^a, \widetilde{s_2^a}, s_3^a\rangle$$

  where $\widetilde{s_2^a} = H(k_{\mathsf{ch}}\|a)$ and (s_1^a, s_3^a) are computed honestly using the witness w.

  We briefly comment on the relation between H', which is used by the simulator, and H'_1. Both oracles can be viewed as the same parameterized oracle, where the parameters are every s_1^a. In H'_1, the s_1^a are honestly generated, whereas in H' they are generated by the HVZK simulator for Σ.

- $\underline{\mathsf{Hyb}_2 = \mathsf{Sim}}$: The only differences from Hyb_1 are for each $a \in A\backslash\{0\}$, we replace (s_1^a, s_3^a) with an honest-verifier simulated transcript $\left(\widetilde{s_1^a}, \widetilde{s_3^a}\right)$ from

$$\left(\widetilde{s_1^a}, \widetilde{s_2^a}, \widetilde{s_3^a}\right) = \mathsf{Sim}_{\Sigma}(x, H(k_{\mathsf{ch}}\|a); H(k\|a))$$

  and update the random oracle accordingly, as defined in eq. (22.2) with $s_1^a = \widetilde{s_1^a}$. This results in the oracle H'.

  In the security game, this modification results in a NIZK

$$|\widetilde{\pi}\rangle \propto \sum_{a\in A\backslash\{0\}} (-1)^{a\cdot s} |a\rangle \otimes |\widetilde{s_1^a}, \widetilde{s_2^a}, \widetilde{s_3^a}\rangle$$

  Note that the definition of $\left(\widetilde{s_1^a}, \widetilde{s_3^a}\right)$ is implicit, and they are only computed coherently as necessary: (1) in answering queries to H'_2, (2) in computing $|\widetilde{\pi}\rangle$, and (3) in applying the isometry $|a\rangle \otimes |y\rangle \mapsto |a\rangle \otimes |y \oplus \left(\widetilde{s_1^a}, \widetilde{s_2^a}, \widetilde{s_3^a}\right)\rangle$ to check the deletion certificate.

[1] Note that queries to H'_1 can be answered lazily, preventing an exponential blow-up from reprogramming an exponential number of positions. Since H'_1 is implicitly pre-defined in terms of H, which is fixed, this does *not* require maintaining state, e.g. through the compressed oracle technique.

[2] We abstract the expansion of s_1^a out of the definition of H'_1 to emphasize that s_1^a is treated as an implicitly defined parameter to the reprogrammed oracle.

We now show in Claims 22.1.2 to 22.1.4 that

$$\begin{aligned}(\mathsf{Hyb}_0, O_H) &\approx_c \left(\mathsf{Hyb}_1, O_{H_1'}\right)\\ &\approx_c (\mathsf{Hyb}_2, O_{H'})\\ &\approx_c (\mathsf{Hyb}_2, O_H)\end{aligned}$$

over the choice of the random oracle H.

Claim 22.1.2.

$$\{(\mathsf{Hyb}_0, O_H) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\} \approx \left\{\left(\mathsf{Hyb}_1, O_{H_1'}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\}$$

Proof. Consider the sub-hybrid experiment Hyb_0':

- $\underline{\mathsf{Hyb}_0'}$: This hybrid is identical to Hyb_0, except that H is replaced with a reprogrammed oracle H_0', defined as
$$H_0'(v) \coloneqq \begin{cases} G(v') & \text{if } v = k_{\mathsf{ch}} \| v' \text{ for some } v' \\ H(v) & \text{else.}\end{cases}$$
where $\mathcal{X} = \{0,1\}^\lambda \times \mathcal{X}_2$ and $G \leftarrow \mathsf{Func}(\mathcal{X}_2, \mathcal{Y})$ is a fresh random oracle.
- $\underline{\mathsf{Hyb}_1}$: The only difference between Hyb_0' and Hyb_1 is the challenger samples a key $k_{\mathsf{ch}} \leftarrow \{0,1\}^\lambda$ and G is defined by $G(v) = H(k_{\mathsf{ch}} \| v)$.

Since G is a truly random function and is only used in the definition of H_0',

$$\{(\mathsf{Hyb}_0, O_H) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\} = \left\{\left(\mathsf{Hyb}_0', O_{H_0'}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\}$$

Furthermore, lemma 20.1.2 implies[3]

$$\left\{\left(\mathsf{Hyb}_0', O_{H_0'}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\} \approx_c \left\{\left(\mathsf{Hyb}_1, O_{H_1'}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\}$$

□

Claim 22.1.3. *If* Sigma *is a sigma-protocol for a language* $\mathcal{L}$ *with* $2^{-\lambda}$*-security, where* $\nu(\lambda)$ *is a negligible function, then*

$$\left\{\left(\mathsf{Hyb}_1, O_{H_1'}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\} \approx \{(\mathsf{Hyb}_2, O_{H'}) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\}$$

Proof. Let a_j be the j'th lexicographically least element of A. We define a series of $2 \cdot |A| = 2 \cdot 2^{\lambda/2}$ intermediate hybrid experiments $\mathsf{Hyb}_{1,i,1}$ and $\mathsf{Hyb}_{1,i,2}$ for $i \in [|A|]$, as well as an additional hybrid experiment Hyb_1'.

- $\underline{\mathsf{Hyb}_1'}$: This hybrid is identical to Hyb_1, except that the challenger uses an additional random oracle $G \leftarrow \mathsf{Func}(\{0,1\}^\lambda, \mathcal{Y})$. Whenever it would evaluate $H(k\|v)$ for some v, it instead (coherently) evaluates $G(v)$.

[3] Since G is used to define H_0', changing G to match $H(k_{\mathsf{ch}}\|\cdot)$ causes H_0' to become H_1'.

- $\underline{\mathsf{Hyb}_{1,i,1}}$: This identical to Hyb'_1, except for the following changes. First, for every $j < i$, the honestly computed $(s_1^{a_j}, s_3^{a_j})$ are replaced by $(\widetilde{s_1^{a_j}}, \widetilde{s_3^{a_j}})$, which are generated by Sim_Σ using randomness $G(a_j)$. Since $s_1^{a_j}$ is modified, the random oracle is also updated accordingly to become $H'_{1,i}$, as defined by eq. (22.2) with $s_1^{a_j} := \widetilde{s_1^{a_j}}$ for $j < i$.

  The second difference is purely syntactic. Instead of getting access to G, the challenger gets access to G_{a_i}, which is identical to G except that $G_{a_i}(a_i) = \bot$. Additionally, it receives a classical copy of $(s_1^{a_i}, s_3^{a_i})$, which are computed by the honest Σ prover using (x, w) and randomness $G(a_i)$.

- $\underline{\mathsf{Hyb}_{1,i,2}}$: This is identical to $\mathsf{Hyb}_{1,i,1}$, except that the honestly computed $(s_1^{a_i}, s_3^{a_i})$ are replaced by $(\widetilde{s_1^{a_i}}, \widetilde{s_3^{a_i}})$, where
$$\left(\widetilde{s_1^{a_i}}, \widetilde{s_2^{a_i}}, \widetilde{s_3^{a_i}}\right) = \mathsf{Sim}_\Sigma(x, H(k_{\mathsf{ch}} \| a_i); G(a_i))$$

- $\underline{\mathsf{Hyb}_2}$: The only difference between Hyb_2 and $\mathsf{Hyb}_{1,i_{\max},2}$, where $i_{\max} = 2^{\lambda/2}$, is the challenger samples a key $k \leftarrow \{0,1\}^\lambda$ and G is defined by $G(v) = H(k\|v)$.

lemma 20.1.2 implies

$$\left\{\left(\mathsf{Hyb}_1, O_{H'_1}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\} \approx_c \left\{\left(\mathsf{Hyb}'_1, O_{H'_1}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\}$$

Observe that $\mathsf{Hyb}_{1,0,1}$ makes only syntactic changes from Hyb'_1, so for all H,

$$\mathsf{Hyb}'_1 = \mathsf{Hyb}_{1,0,1}$$

The same observation holds for every $\mathsf{Hyb}_{1,i,2}$ and $\mathsf{Hyb}_{1,i+1,1}$, so for all H,

$$\mathsf{Hyb}_{1,i,2} = \mathsf{Hyb}_{1,i+1,1}$$

We now show that[4]

$$\{(\mathsf{Hyb}_{1,i,1}, O_{H'_{1,i}}) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\} \approx_c^{2^{-\lambda}} \{(\mathsf{Hyb}_{1,i,2}, O_{H'_{1,i+1}}) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\}$$

by reducing to the $2^{-\lambda}$-security of Σ. The reduction simulates the random oracles H and G_{a_i} using the compressed oracle technique [Zha19a],[5] then declares $H(k_{\mathsf{ch}}\|a_i)$ as its challenge $s_2^{a_i}$ in an honest-verifier execution of Σ. It receives either $(s_1^{a_i}, s_3^{a_i})$ generated by an honest prover, or $(\widetilde{s_1^{a_i}}, \widetilde{s_3^{a_i}})$ generated by the HVZK simulator Sim_Σ. Using these, it can compute the rest of the experiment according to the description of $\mathsf{Hyb}_{1,i,1}$. In the former case, the resulting distribution is $\mathsf{Hyb}_{1,i,1}$; in the latter, it is $\mathsf{Hyb}_{1,i,2}$. Therefore, if the two distributions above were distinguishable with advantage ϵ, then applying that same distinguisher would distinguish an honest Σ prover's messages from the output of the honest-verifier simulator Sim_Σ also with advantage ϵ. The $2^{-\lambda}$-security of Σ implies that $\epsilon \leq 2^{-\lambda}$ for all QPT distinguishers.

lemma 20.1.2 implies

$$\left\{\left(\mathsf{Hyb}_{1,i_{\max},2}, O_{H'_{1,i+1}}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\} \approx_c \{(\mathsf{Hyb}_2, O_{H'}) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\}$$

By the triangle inequality, the distinguishing advantage between $\left\{\left(\mathsf{Hyb}_1, O_{H'_1}\right) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\right\}$ and $\{(\mathsf{Hyb}_2, O_{H'}) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\}$ for any QPT distinguisher is bounded by $\mathsf{negl}(\lambda) + 2^{\lambda/2} \cdot 2^{-\lambda} + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$.

□

[4] As before, $H_{1,i}$ is defined using $s_1^{a_i}$, so modifying it to become simulated changes $H'_{1,i}$ to $H'_{1,i+1}$.

[5] If Σ is HVZK in the QROM, e.g. because it is designed for the QROM, then this step is unnecessary.

Claim 22.1.4.

$$\{(\mathsf{Hyb}_2, O_{H'}) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\} \approx \{(\mathsf{Hyb}_2, O_H) : H \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})\}$$

Proof. We reduce to the direct product hardness of subspace states (lemma 2.3.3). Assume, for the sake of contradiction, that some QPT distinguisher D with auxiliary input register $\mathsf{R_D}$ were able to distinguish between these two distributions.

Before describing the reduction, we claim that D has noticeable advantage conditioned on Hyb_2 not outputting $\perp$, i.e. conditioned on the simulator's certificate check passing. We also claim that Hyb_2 outputs $\perp$ with at most $1 - 1/p$ probability for some polynomial p. We reduce these claims to direct product hardness (lemma 2.3.3). The reduction takes in a subspace state $|A\rangle$ and membership oracles O_A, $O_{A^\perp}$. If either of the claimed conditions do not hold, then D must have noticeable advantage conditioned on Hyb_2 outputting $\perp$. By lemma 20.1.1, measuring one of D's queries at random produces a point x^* where $H(x^*) \neq H'(x^*)$ with noticeable probability. These are exactly values of the form $a \| b$ for $a \in A \backslash \{0\}$. Since H' can be implemented using O_A, this procedure finds an element of A with noticeable probability using only a polynomial number of queries to O_A, without using $|A\rangle$. Combining this with a measurement of $|A\rangle$ in the Hadamard basis, the reduction obtains a pair of vectors in $A\backslash\{0\} \times A^\perp\backslash\{0\}$ with noticeable probability, breaking direct product hardness.

Next, we show how to break direct product hardness of subspace states using a D which has noticeable distinguishing advantage between $(\mathsf{Hyb}_2, O_{H'})$ and (Hyb_2, O_H), conditioned on Hyb_2 not outputting $\perp$. The reduction takes as input a random subspace state $|A\rangle$ and oracle access to membership oracles O_A, $O_{A^\perp}$. It runs the simulator (i.e. Hyb_2) using $|A\rangle$. Whenever it would check that a vector v is in A (respectively, $A^\perp$), it queries v to O_A (respectively, $O_{A^\perp}$). To implement step 6 (the certificate check), it first applies the following operations to register $\mathsf{R_A}$: (1) a Hadamard operation, (2) the isometry $|y\rangle \mapsto |y - s\rangle$, and (3) a Hadamard operation. Then, it uses O_A and $O_{A^\perp}$ to implement the PVM $\{|A\rangle\langle A|, I - |A\rangle\langle A|\}$ on register $\mathsf{R_A}$, as described by lemma 2.3.2. If the result of the simulator is $\mathsf{R}_\mathcal{A} \neq \perp$, then it runs the distinguisher $D^{H'}(\mathsf{R_D}, \mathsf{R}_\mathcal{A})$ and measures a random query that D makes to H'. Let $q = q_1 \| q_2$ be the query measurement result. Finally, the reduction measures $\mathsf{R_A}$ in the Hadamard basis to obtain a value v and outputs (q_1, v).

Recall that Hyb_2 outputs $\mathsf{R}_\mathcal{A} \neq \perp$ with noticeable probability. Condition on this case occurring. This happens exactly when the measurement on $\mathsf{R_A}$ returns result $|A\rangle\langle A|$, in which case $\mathsf{R_A}$ collapses to $|A\rangle$. Therefore the measurement result $v \in A^\perp$ and is not 0 with overwhelming probablity. Furthermore, we previously established that D has noticeable advantage in this case; therefore lemma 20.1.1 implies that $H(q_1\|q_2) \neq H'(q_1\|q_2)$ with noticeable probability. Whenever this occurs, $q_1 \in A$. Therefore $(q_1, v) \in A\backslash\{0\} \times A^\perp\backslash\{0\}$ with noticeable probability, violating lemma 2.3.3. □

□

Chapter 23

Negative Results

In this section, we give evidence that certifiable deniability is in fact *impossible* in the plain model. Specifically, we show a black-box barrier against obtaining signatures with certified deniability in the plain model; any work achieving this must use non-black-box techniques in their security proof.

23.1 Distinguishing Between Unitary Oracles.

Before proving the main result of this section, we introduce a supporting lemma that generalizes a result from [BBBV97] about reprogramming quantum-accessable oracles. This lemma may be of independent interest.

In their original version, they consider a quantum adversary who has quantum query access to one of two classical oracles H and H'. They bound the ability of the adversary to distinguish between the two oracles in terms of the amplitude with which it queries on (classical) inputs x where $H(x) \neq H'(x)$. As a simple corollary, if the adversary is able to distinguish the two oracles in a polynomial number of queries, then measuring one of its queries at random produces an x such that $H_0(x) \neq H_1(x)$ with noticeable probability.

We observe that a similar statement holds for quantum oracles, i.e. oracles which take in a quantum input, perform quantum computation, and produce quantum output. If the adversary is able to distinguish the two oracles in a polynomial number of queries, then outputting the query register at a random query produces a mixed state with noticeable probability mass on states $|\psi\rangle$ where $H(|\psi\rangle)$ is far from $H'(|\psi\rangle)$.

Lemma 23.1.1. *Let H and H' be oracle-accessible unitaries and let $A^{(\cdot)}$ be a quantum oracle algorithm with auxiliary input $|\psi_0\rangle$. Let*

$$|\psi_t\rangle = \sum_i \alpha_{t,i} \, |\phi_{t,i}\rangle_{\mathsf{R_A}} \otimes |q_{t,i}\rangle_{\mathsf{R_Q}}$$

be a Schmidt decomposition of the state of A^H on submitting query t, where $\mathsf{R_A}$ is the internal register of A and $\mathsf{R_Q}$ is the query register. Let ψ'_t similarly be the state of $A^{H'}$ on submitting query t. Then for all $T \in \mathbb{N}$,

$$\mathsf{TD}[|\psi_{T+1}\rangle \langle\psi_{T+1}| \, , |\psi'_{T+1}\rangle \langle\psi'_{T+1}|] \leq \sqrt{4T \sum_{t=1}^{T} \sum_i |\alpha_{t,i}|^2 \mathsf{TD}[H(|q_{t,i}\rangle), H'(|q_{t,i}\rangle)]^2}$$

Proof. The oracle algorithm A^H can be expressed as sequence of unitaries A_t interleaved with unitary

oracle operations H, i.e.

$$|\psi_{T+1}\rangle = A_{T+1}\left(\prod_{t=1}^{T} HA_t\right)|\psi_0\rangle$$

Note that $|\psi_{t+1}\rangle$ is prepared by applying A_{t+1} to the result $H|\psi_t\rangle$ of the prior query. Define $|E_t\rangle$ to be the error term resulting from answering the t'th query of A^H using H' instead of H, i.e.

$$|E_t\rangle = H'|\psi_t\rangle - H|\psi_t\rangle$$

Then we can express the state of $A^H(|\psi_0\rangle)$ on submitting the $(T+1)$'th query (before it is answered) as

$$|\psi_{T+1}\rangle = A_T H|\psi_T\rangle \tag{23.1}$$
$$= A_T(H'|\psi_T\rangle - |E_T\rangle) \tag{23.2}$$

Applying this decomposition recursively on $|\psi_T\rangle$ yields

$$|\psi_{T+1}\rangle = A_{T+1}\left(\prod_{t=1}^{T} H'A_t\right)|\psi_0\rangle - \sum_{t=1}^{T}\left(\prod_{i=t+1}^{T} HA_i\right)A_t|E_t\rangle \tag{23.3}$$
$$= |\psi'_{T+1}\rangle - \sum_{t=1}^{T}\left(\prod_{i=t+1}^{T} HA_i\right)|E_t\rangle \tag{23.4}$$

Thus we have

$$\mathsf{TD}[|\psi_{T+1}\rangle\langle\psi_{T+1}|, |\psi'_{T+1}\rangle\langle\psi'_{T+1}|]$$
$$= \sqrt{1 - |\langle\psi_T|\psi'_T\rangle|^2} \tag{23.5}$$
$$\leq \sqrt{1 - |\langle\psi_T| + (\langle\psi'_T| - \langle\psi_T|)|\psi_T\rangle|^2} \tag{23.6}$$
$$\leq \sqrt{|(\langle\psi'_T| - \langle\psi_T|)|\psi_T\rangle|^2} \tag{23.7}$$
$$\leq \sqrt{\||\psi'_T\rangle - |\psi_T\rangle\|_2^2} \tag{23.8}$$
$$= \sqrt{\left\|\sum_{t=1}^{T}\left(\prod_{i=t+1}^{T} HA_i\right)|E_t\rangle\right\|_2^2} \tag{23.9}$$
$$\leq \sqrt{T\sum_{t=1}^{T}\left\|\left(\prod_{i=t+1}^{T} HA_i\right)|E_t\rangle\right\|_2^2} \tag{23.10}$$
$$= \sqrt{T\sum_{t=1}^{T}\||E_t\rangle\|_2^2} \tag{23.11}$$
$$= \sqrt{T\sum_{t=1}^{T}\|H'|\psi_t\rangle - H|\psi_t\rangle\|_2^2} \tag{23.12}$$
$$= \sqrt{T\sum_{t=1}^{T}\left|\sum_{i,j}\alpha_{t,i}\overline{\alpha_t, j}\,\langle\phi_{t,i}\rangle\phi_{t,j}\langle q_{t,i}|(H-H')^\dagger(H-H')|q_{t,j}\rangle\right|^2} \tag{23.13}$$

$$\leq \sqrt{T \sum_{t=1}^{T} \sum_{i} |\alpha_{t,i}|^2 \|(H - H') |q_{t,i}\rangle\|_2^2} \tag{23.14}$$

$$\leq \sqrt{T \sum_{t=1}^{T} \sum_{i} |\alpha_{t,i}|^2 \|(H - H') |q_{t,i}\rangle\|_1^2} \tag{23.15}$$

$$= \sqrt{4T \sum_{t=1}^{T} \sum_{i} |\alpha_{t,i}|^2 \mathsf{TD}[H(|q_{t,i}\rangle), H' |q_{t,i}\rangle]^2} \tag{23.16}$$

where eq. (23.8) follows from Cauchy-Schwarz; eq. (23.10) follows from Jensen's inequality; eq. (23.11) follows from the invariance of the ℓ_2 norm under unitaries; eq. (23.14) follows from Jensen's inequality. □

Corollary 23.1.2 (One-Way to Hiding for Unitary Oracles)**.** *Let H and H' be oracle-accessible unitaries and let $A^{(\cdot)}$ be a quantum oracle algorithm with auxiliary input $|\psi_0\rangle$ that uses at most T queries. Denote the mixed state resulting from measuring A^H's query register at a random time t as*

$$\frac{1}{T} \sum_{t=1}^{T} |t\rangle \langle t| \otimes \sum_{i} |\alpha_{t,i}|^2 |q_{t,i}\rangle \langle q_{t,i}|$$

For $0 < \delta \leq 1$, denote the set S_δ as being the subset of (t, i) pairs such that $\mathsf{TD}[H(|q_{t,i}\rangle), H'(|q_{t,i}\rangle)] \geq \delta$. If $A^{(\cdot)}$ distinguishes between H and H' with advantage ϵ in T queries, then

$$\frac{1}{T} \sum_{(t,i) \in S_\delta} |\alpha_{t,i}|^2 \geq \frac{\frac{\epsilon^2}{4T^2} - \delta^2}{1 - \delta^2}$$

If ϵ were noticeable, T were polynomial, and δ were set to $(1/2) \cdot \epsilon/(2T)$, then this immediately shows that the probability mass on pure states where H and H' differ by a noticeable trace distance is noticeable.

23.2 Plain Model Black-Box Barrier

For our black-box barrier, we model the simulator as having access to the adversary's unitary dilation and its inverse. However, we do not allow the simulator direct access to the adversary's internal registers, including its auxiliary input. As we discuss later, this is equivalent to a model where the simulator can directly access the internal registers, but they are ideally obfuscated.

Theorem 23.2.1. *There do not exist signatures with certified deniability in the plain model whose security proof makes only black-box use of the adversary's unitary dilation and its inverse.*

Remark 23.2.2. *We note that a similar statement holds for NIZKs with certified deniability as well. The proof is almost identical, so we omit it here.*

Proof. Let $\mathsf{Sig\text{-}CDen} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Del}, \mathsf{DelVer})$ be a candidate signature scheme with a black-box proof of certified deniability, i.e. which gives a simulator Sim that uses the adversary's unitary dilation as a black-box. We give a QPT adversary Adv together with a QPT distinguisher D that distinguishes Sim from the real experiment.

Adv receives as auxiliary input a post-quantum signing key $\widetilde{k}$, which we denote as $\mathsf{Adv}_{\widetilde{k}}$. During the certified deniability game, $\mathsf{Adv}_{\widetilde{k}}$ receives a verification key vk, a state $|\sigma\rangle$, and a message m, then runs $\mathsf{Verify}(\mathsf{vk}, |\sigma\rangle, m)$, and if it accepts, outputs a post-quantum signature $\sigma^{(\widetilde{k})}_{m,\mathsf{vk}} \leftarrow \mathsf{PQSig.Sign}(\widetilde{k}, \mathsf{vk}\|m)$ on the verification key concatenated with the message. It measures the result of this to get $\sigma^{(\widetilde{k})}_{m,\mathsf{vk}}$, which it writes to register $\mathsf{R}_{\mathcal{A}}$, then uncomputes the program. By the correctness of $\mathsf{Sig\text{-}CDen}$, this is gentle. Finally, it runs $\mathsf{Del}(|\sigma_m\rangle)$ to obtain a valid deletion certificate in register $\mathsf{R}_{\mathsf{cert}}$ and outputs $\mathsf{R}_{\mathsf{cert}}$ along with $\mathsf{R}_{\mathcal{A}}$.

The distinguisher receives as auxiliary input the corresponding post-quantum verification key $\widetilde{\mathsf{vk}}$. On input $\mathsf{R}_{\mathcal{A}}$, it runs the post-quantum signature verification $\mathsf{PQSig.Verify}(\widetilde{\mathsf{vk}}, \mathsf{R}_{\mathcal{A}})$ to check if $\mathcal{A}$ obtained a signature on $\mathsf{vk}\|m$. If so, it outputs 1, and otherwise, outputs 0.

We now show that this adversary and distinguisher pair achieves noticeable advantage against any simulator Sim. Observe that in the real certified deniability experiment, D outputs 1 with overwhelming probability, due to the correctness of the signature and obfuscation schemes. Conversely, we claim that against any QPT simulator, D outputs 0 with overwhelming probability.

Consider the following sequence of hybrid experiments:

- $\underline{\mathsf{Hyb}_0}$: The simulated certified deniability experiment. In more detail, sample $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Run $\mathsf{Sim}^{\mathsf{Adv}_{\widetilde{k}}}$ to obtain an output register $\mathsf{R}_{\mathcal{A}}$.
- $\underline{\mathsf{Hyb}_1}$: This experiment is the same as Hyb_0, except that $\mathsf{Adv}_{\widetilde{k}}$ is replaced by $\mathsf{Adv}_\perp$, which immediately honestly deletes the signature and always outputs $\perp$ in register $\mathsf{R}_{\mathcal{A}}$.

$\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$ from the unforgeability of the candidate signature scheme; otherwise, corollary 23.1.2 implies that outputting $\mathsf{Sim}^{\mathsf{Adv}_{\widetilde{k}}}$'s query register on a random query produces a mixed state with noticeable probability mass on an input $|\psi\rangle$ where $\mathcal{A}_{\widetilde{k}}(|\psi\rangle)$ and $\mathcal{A}_\perp(|\psi\rangle)$ are noticeably far, which is is exactly the set of points where $|\psi\rangle$ passes signature verification with noticeable probability.

We now claim that the original distinguisher $D(\widetilde{\mathsf{vk}})$ outputs 0 with overwhelming probability in Hyb_1. Otherwise, $\mathsf{Sim}^{\mathcal{A}_\perp}$ must output a valid signature under $\widetilde{\mathsf{vk}}$, which it is independent of, which would violate the unforgeability of the post-quantum signature scheme. Since Hyb_1 is indistinguishable from Hyb_0 for all QPT distinguishers, $D(\widetilde{\mathsf{vk}})$ must also output 0 with overwhelming probability in Hyb_0, which is simply the output distribution of the certified deniability simulator.

□

Obfuscated Auxiliary Input. One could also consider implementing the adversary described above using an obfuscated program which $\mathcal{A}$ receives as auxiliary input. If the obfuscation was ideal, then even allowing the simulator direct access to the adversary's auxiliary input would be equivalent to the model above. Even with a weaker form of obfuscation, it still might not be possible to extract a signature from the obfuscated program even using non-black-box techniques.

Bibliography

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242, 2009.

[ABF+19] Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 441–471. Springer, Cham, May 2019.

[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.

[ABKK23] Amit Agarwal, James Bartusek, Dakshita Khurana, and Nishant Kumar. A new framework for quantum oblivious transfer. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 363–394. Springer, Cham, April 2023.

[AC02] Mark Adcock and Richard Cleve. A quantum goldreich-levin theorem with cryptographic applications. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002.

[AC12a] 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.

[AC12b] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 41–60. ACM Press, May 2012.

[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, Berlin, Heidelberg, August 2015.

[AK21] Prabhanjan Ananth and Fatih Kaleoglu. Unclonable encryption, revisited. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 299–329. Springer, Cham, November 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.

[AK24] Kasra Abbaszadeh and Jonathan Katz. Non-interactive zero-knowledge proofs with certified deletion. *IACR Cryptol. ePrint Arch.*, page 1848, 2024.

[AKL23] Prabhanjan Ananth, Fatih Kaleoglu, and Qipeng Liu. Cloning games: A general framework for unclonable primitives. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 66–98. Springer, Cham, August 2023.

[AKN+23] Shweta Agrawal, Fuyuki Kitagawa, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Public key encryption with secure key leasing. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 581–610. Springer, Cham, April 2023.

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 501–530. Springer, Cham, October 2021.

[APV23] Prabhanjan Ananth, Alexander Poremba, and Vinod Vaikuntanathan. Revocable cryptography from learning with errors. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 93–122. Springer, Cham, November / December 2023.

[BB84] Charles H Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, page 175–179, 1984.

[BBBV97] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[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, Berlin, Heidelberg, February 2014.

[BF10] Niek J. Bouman and Serge Fehr. Sampling in a quantum population, and applications. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 724–741. Springer, Berlin, Heidelberg, August 2010.

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Berlin, Heidelberg, August 1993.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001.

[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, Berlin, Heidelberg, March 2014.

[BGK+24] James Bartusek, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Software with certified deletion. In Marc Joye and Gregor Leander, editors,

EUROCRYPT 2024, Part IV, volume 14654 of *LNCS*, pages 85–111. Springer, Cham, May 2024.

[BI20a] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 92–122. Springer, 2020.

[BI20b] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 92–122. Springer, Cham, November 2020.

[BK23] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 192–223. Springer, Cham, August 2023.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 671–684. ACM Press, June 2018.

[BL90] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, New York, August 1990.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

[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, Berlin, Heidelberg, March 2017.

[BR24] James Bartusek and Justin Raizes. Secret sharing with certified deletion. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 184–214. Springer, 2024.

[BS20] Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 269–279. ACM Press, June 2020.

[BS23] Shalev Ben-David and Or Sattath. Quantum tokens for digital signatures. *Quantum*, 7:901, 2023.

[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, Berlin, Heidelberg, December 2013.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Berlin, Heidelberg, August 1997.

[CGV22] Andrea Coladangelo, Shafi Goldwasser, and Umesh V. Vazirani. Deniable encryption in a quantum world. In Stefano Leonardi and Anupam Gupta, editors, *54th ACM STOC*, pages 1378–1391. ACM Press, June 2022.

[CLLZ21a] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 556–584, Virtual Event, August 2021. Springer, Cham.

[CLLZ21b] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 556–584, Cham, 2021. Springer International Publishing.

[CMSZ22] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments: Breaking the quantum rewinding barrier. In *62nd FOCS*, pages 49–58. IEEE Computer Society Press, February 2022.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, New York, August 1990.

[DFL+09] Ivan Damgård, Serge Fehr, Carolin Lunemann, Louis Salvail, and Christian Schaffner. Improving the security of quantum protocols via commit-and-open. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 408–427. Springer, Berlin, Heidelberg, August 2009.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Cham, August 2019.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th ACM STOC*, pages 409–418. ACM Press, May 1998.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[Gao03] Shuhong Gao. *A New Algorithm for Decoding Reed-Solomon Codes*, pages 55–68. Springer US, Boston, MA, 2003.

[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.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 82–115. Springer, Cham, November 2017.

[GMR24] Vipul Goyal, Giulio Malavolta, and Justin Raizes. Unclonable commitments and proofs. In *22nd Theory of Cryptography Conference 2024, TCC 2024, Dec 2-6, 2024, Bocconi University, Milan, Italy*, 2024.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.

[HKM+24] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption, and more. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 434–456. Springer, Cham, May 2024.

[HMNY22] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting zero-knowledge proof for QMA. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 239–268. Springer, Cham, August 2022.

[Hoe94] Wassily Hoeffding. *Probability Inequalities for sums of Bounded Random Variables*, pages 409–426. Springer New York, New York, NY, 1994.

[ISN87] M. Ito, A. Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Proc. IEEE Global Telecommunication Conf. (Globecom'87)*, pages 99–102, 1987.

[JLLW23] Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. The pseudorandom oracle model and ideal obfuscation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 233–262. Springer, Cham, August 2023.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Cham, May / June 2022.

[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, Berlin, Heidelberg, February / March 2013.

[LM01] J.L. Lagrange and T.J. McCormack. *Lectures on Elementary Mathematics*. Open Court Publishing Company, 1901.

[Lut10] Andrew Lutomirski. An online attack against wiesner's quantum money, 2010.

[LV18] Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 699–708, New York, NY, USA, 2018. Association for Computing Machinery.

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum Fiat-Shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 326–355. Springer, Cham, August 2019.

[MPY24] Tomoyuki Morimae, Alexander Poremba, and Takashi Yamakawa. Revocable quantum digital signatures. In Frédéric Magniez and Alex Bredariol Grilo, editors, *19th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2024, September 9-13, 2024, Okinawa, Japan*, volume 310 of *LIPIcs*, pages 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Berlin, Heidelberg, August 2003.

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In *ITCS 2023*, pages 90:1–90:14. LIPIcs, January 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, Cham, August 2019.

[RK05] Renato Renner and Robert König. Universally composable privacy amplification against quantum adversaries. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 407–425. Springer, Berlin, Heidelberg, February 2005.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[Shm22] Omri Shmueli. Semi-quantum tokenized signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 296–319. Springer, Cham, August 2022.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Berlin, Heidelberg, April 2012.

[Unr16a] Dominique Unruh. Collapse-binding quantum commitments without random oracles. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 166–195. Springer, Berlin, Heidelberg, December 2016.

[Unr16b] Dominique Unruh. Computationally binding quantum commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 497–527. Springer, Berlin, Heidelberg, May 2016.

[VZ21] Thomas Vidick and Tina Zhang. Classical proofs of quantum knowledge. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 630–660. Springer, Cham, October 2021.

[WB86] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, jan 1983.

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999.

[WZ82] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, Oct 1982.

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

[Zha19a] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Cham, August 2019.

[Zha19b] 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, Cham, May 2019.

[Zha23] Mark Zhandry. Tracing quantum state distinguishers via backtracking. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 3–36. Springer, Cham, August 2023.